UNIVERSITÀ DI PISA

FACOLTÀ DI INGEGNERIA
Corso di Laurea Specialistica in Ingegneria Informatica per la Gestione d'Azienda

TESI DI LAUREA SPECIALISTICA

BPMN e YAWL a confronto: due approcci al BPM su un caso di studio reale

Candidato

Sinibaldi Fabio

Relatori

Prof. Andrea Tomasi

Dott. Alessio Bechini

ANNO ACCADEMICO 2008 – 2009

"Senza deviazione dalla norma, il progresso non è possibile."

Frank Zappa (1940-1993), musicista americano.

Sommario

Indice delle figure

Abstract

Nell'ambito del BPM vi è la pratica ormai consolidata di utilizzare il linguaggio BPMN per la mappatura dei processi e successivamente tradurre il modello utilizzando un linguaggio eseguibile quale il BPEL. Scopo di questa tesi è quello di fare luce sui punti di forza e di debolezza di tale approccio e confrontarli con un'alternativa di recente realizzazione: YAWL. Tale lavoro di confronto avverrà introducendo prima i due approcci così come vengono descritti e presentati nelle rispettive specifiche e successivamente confrontandoli sul piano teorico / pratico facendo riferimento ad un caso di studio reale su cui verranno applicati. Verranno presi in esame sia i caratteri propri dei linguaggi utilizzati sia le funzionalità offerte dai sistemi che ne supportano l'utilizzo in modo da toccare ogni aspetto rilevante al BPM. Alla fine della trattazione sarà chiaro come le discrepanze concettuali dell'approccio standard precludano la qualità della gestione dei processi e quali sono le idee valide da prendere come esempio dal nuovo approccio proposto dalla YAWL foundation.

Sezione I. Approccio alla modellazione con BPMN ed esecuzione in BPEL

Capitolo 1. Introduzione al BPMN

Il BPMN 1.0 (Business Process Model Notation) (1) (2) è una notazione per la descrizione dei processi di business introdotta nel Maggio 2004 dalla BPMI (Business Process Management Initiative) (3). Lo scopo principale del BPMI (3), ora facente parte del gruppo OMG (Object Management Group) (4) è stato quello di definire un linguaggio standard comprensibile per tutte quelle figure professionali interessate nella modellazione dei processi. In tale ambito infatti vi sono notevoli difficoltà di comunicazione tra analisti sviluppatori e modellatori, con conseguente perdita di efficacia ed efficienza dei processi stessi. Infatti la peculiarità dei vari linguaggi sviluppati per l'esecuzione dei processi di business come ad esempio il BPEL (Business Process for Execution Language) (5) è quella di essere ottimizzati per l'esecuzione, rendendoli così difficilmente comprensibili e scarsamente mantenibili. Per la loro natura quindi non sono adatti ad attività più ad alto livello quale la modellazione dei processi, diventando un ostacolo per la gestione dei processi intra/inter-aziendale. Inoltre ridurre il gap tra i vari utenti della modellazione dei processi riduce il numero di errori nella realizzazione degli stessi dovuti ad incomprensioni delle diverse rappresentazioni utilizzate. Lo standard BPMN (2) definisce una diretta mappatura nel BPEL4WS (BPEL for Web Services) (6) permettendo cosi una automatica traduzione nel linguaggio di esecuzione riducendo ulteriormente il gap.

Il BPMN (1) fornisce il BPD (Business Process Diagram) (7), rappresentazione grafica del modello del processo di business basata sulla tecnica dei diagrammi di flusso, opportunamente adattata alle esigenze del BPM. Nello sviluppare tale notazione infatti il BPMI (3) si è preoccupato di fornire uno strumento che fosse di semplice utilizzo per la modellazione e allo stesso tempo capace di rappresentare la complessità della realtà che si va a rappresentare.

C'è da precisare che il BPMN (1) è estensibile (lo standard prescrive che è possibile definire nuovi elementi al linguaggio purché non in contrasto con quelli già definiti nello standard stesso), e attualmente sono presenti nel mercato diversi dialetti derivati dal linguaggio.

Capitolo 2. Elementi di base del BPD

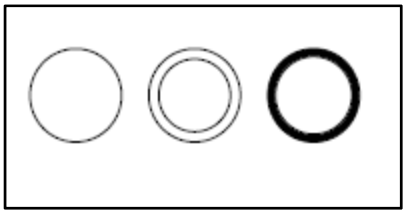
Gli elementi del BPD (7) vengono suddivisi in 4 categorie, per permetterne un facile riconoscimento da parte dell'utente. Come già introdotto possono essere definiti altri simboli per meglio rappresentare la realtà che l'utilizzatore sta modellando.

In BPMN (1) l'informazione rappresentata da ogni elemento del BPD (7)corrisponde ad un set di attributi che lo identifica, ne esplicita le proprietà e la funzione all'interno del modello. Di seguito vengono esposti, suddivisi per categoria, gli elementi standard del BPD (7).

Flow Objects

Sono presenti 3 elementi base per la gestione del flusso d'esecuzione del processo che si modella:

- ***Event*** **-** Un evento è qualcosa che succede durante il corso del processo di business e ne condiziona il flusso e solitamente viene associato ad una causa (***trigger***) o un effetto (***result***). Sono previsti tre tipi di evento, a seconda che questo si verifichi all'inizio (***Start***), durante (***Intermediate***) o alla fine (***End***) del flusso che condizionano.

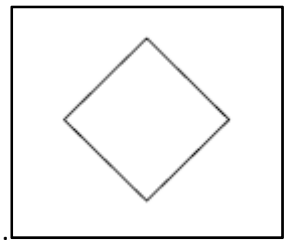


I-1. Start Event, Intermediate Event e End Event

- ***Activity*** – Un'attività è un lavoro effettuato dalla compagnia cui fa riferimento il flusso, e può essere atomica o composta. I tipi di attività previsti sono ***Process***, ***Sub-Process*** e ***Task*** .

I-2. Task

- ***Gateway*** –Un ***gateway*** è utilizzato per controllare la convergenza (***merge*** e ***join***)/ divergenza (***branch*** e ***fork***) dei flussi. All'interno viene indicato il tipo di controllo relativo al ***gateway***

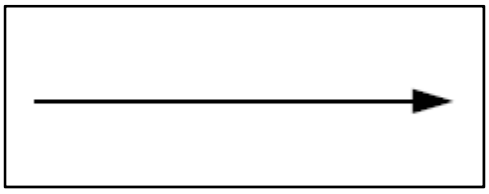


I-3. Gateway

Connecting Objects

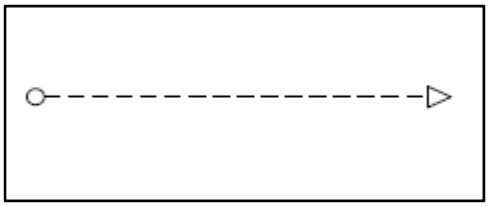
Gli elementi di flusso vengono connessi tra di loro per formare lo scheletro del processo di business dai seguenti elementi :

- ***Sequence Flow*** – Rappresentato da una freccia piena a linea continua, viene utilizzato per collegare ***flow objects*** che avvengono in sequenza.



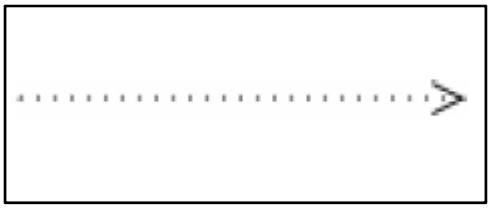
I-4. Sequence Flow

- ***Message Flow*** – Rappresentato da una freccia vuota a linea tratteggiata, viene utilizzato per modellare messaggi che vengono scambiati tra due entità o ruoli associati al processo.



I-5. Message Flow

- ***Association Flow*** – Rappresentato da una linea tratteggiata, serve per associare due elementi (grafici o note) del diagramma. Se viene aggiunta una freccia è utilizzato per indicare una direzione (e.g. input/output di un ***flow object***).

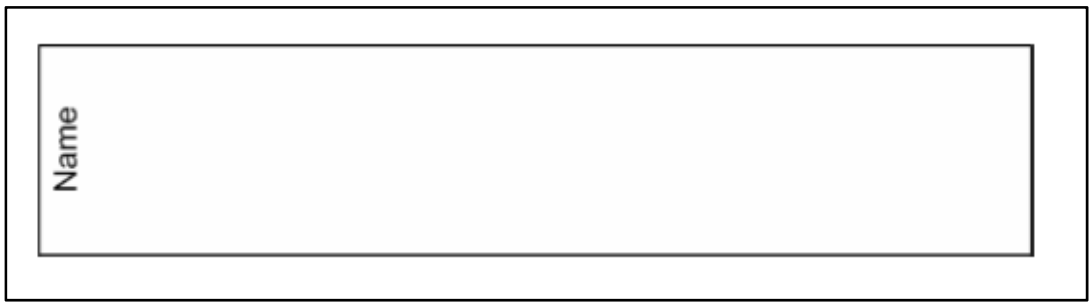


I-6. Association Flow

Swim lanes

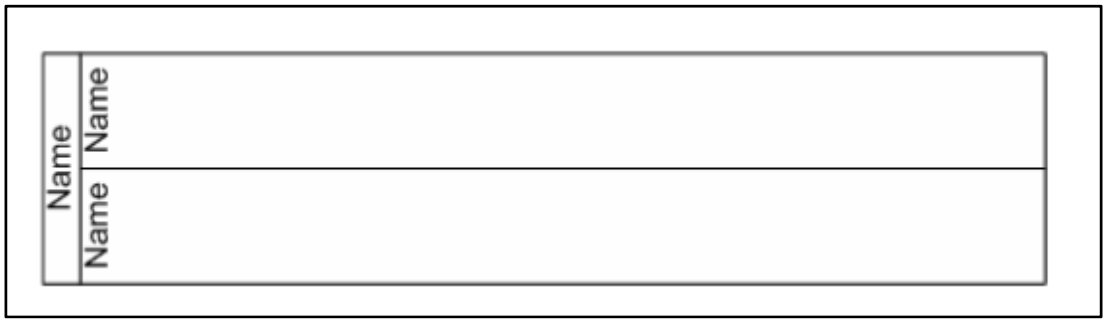
Per organizzare gli elementi del diagramma in diverse categorie (ad esempio enfatizzando diverse funzioni aziendali o diversi ruoli coinvolti nel processo) il BPMN presenta due tipi di ***Swim lane***:

- ***Pool*** – Rappresenta un'entità coinvolta nel processo, come nel caso di due compagnie in ambito B2B o due persone fisiche che collaborano alla sua esecuzione.



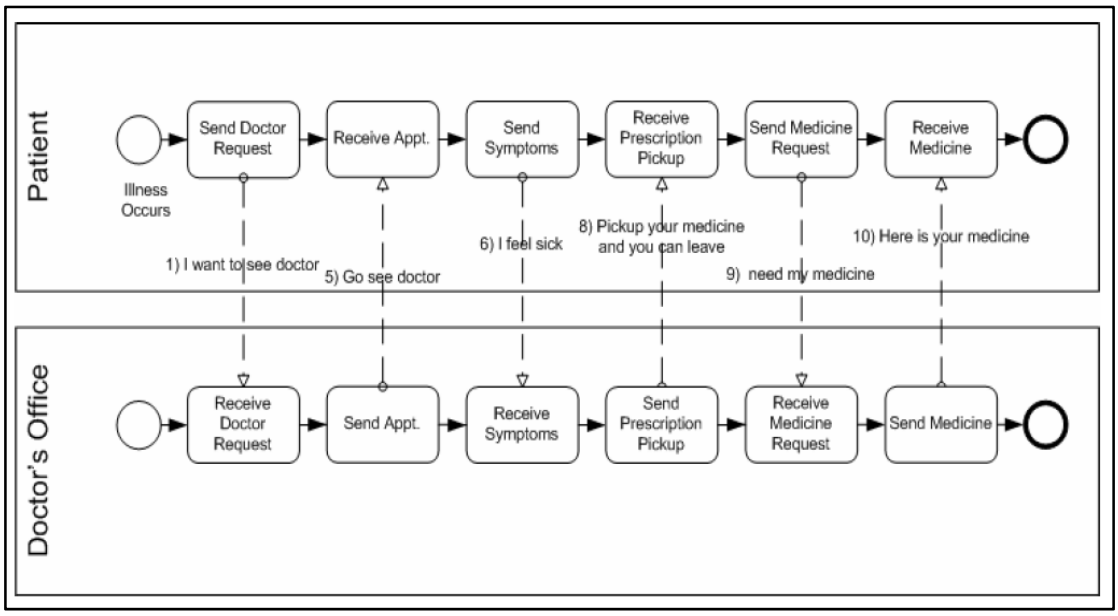
I-7. Pool

- ***Lane*** – Rappresenta una sottosuddivisione del pool (e.g. diverse funzioni di una stessa compagnia).



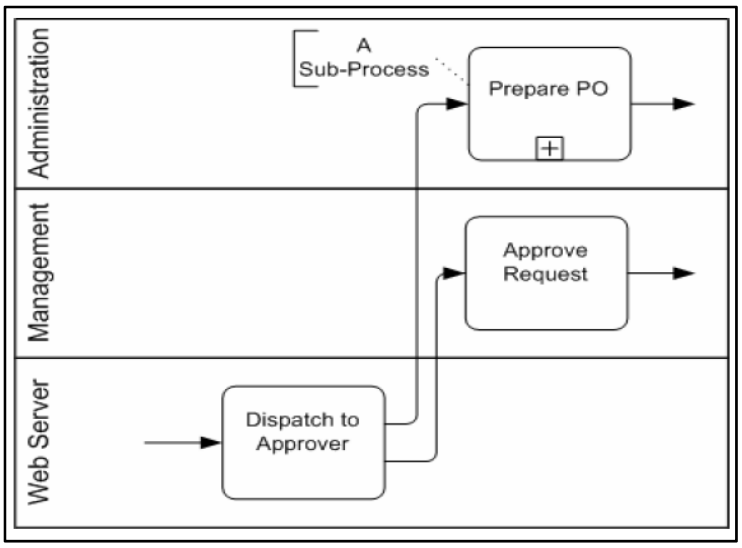
I-8. Lane

In figura I-9 troviamo un primo esempio di BPD. Si vogliono modellare i processi Paziente e Ufficio Medico. In questa modellazione di massima vediamo che il processo paziente inizia con uno ***Star Event*** che si attiva quando si presenta un malessere. A questo punto il paziente comunica con l'ufficio del dottore (il cui processo si attiva incondizionatamente e indipendentemente) attraverso un ***Message Flow***. E' da notare che i ***Pool*** rappresentano entità fisicamente separate e di conseguenza, dato che un processo non può estendersi tra i partecipanti, non è indicato usare dei ***Sequence Flow*** per la sincronizzazione delle attività fra questi ultimi mentre è indicato usare un ***Message Flow***. I due processi vanno avanti scambiandosi dei messaggi fino a che non arrivano all' ***End Event***.



I-9. Message Flow tra due pool

In figura I-10 possiamo analizzare un altro esempio. In questo caso si tratta di alcune attività di un processo di gestione delle richieste da parte di un web server di e-commerce. Attraverso delle ***Lane*** viene individuata la diversa responsabilità delle diverse attività. Essendo il diagramma relativo ad una parte di un'unica entità unica quale la società di e-commerce in questione, la sequenzialità dei ***Task*** e dei ***Sub-Process*** può venire modellata attraverso dei ***Sequence Flow***.

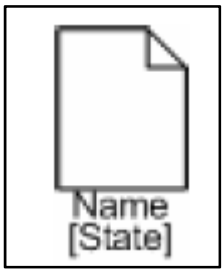


I-10. Sequence Flow tra lane

Artifacts

Il BPD, coerentemente ai propositi di flessibilità e estensibilità del BPMN, permette l'inserimento di diversi elementi predefiniti (artefatti) che meglio permettono di chiarire alcuni aspetti del diagramma. I modelli predefiniti di artefatto sono:

- ***Data Object*** – Si riferisce a dati previsti nel modello, come input o output delle attività.

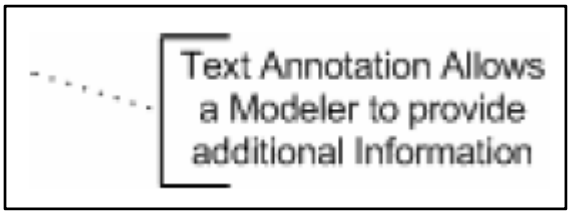


I-11. Data Object

- ***Group*** – Un gruppo, rappresentato da un rettangolo tratteggiato, permette di categorizzare determinati elementi, sia per scopi analitici che per chiarezza del diagramma, senza influenzare il flusso del processo modellato.

I-12. Group

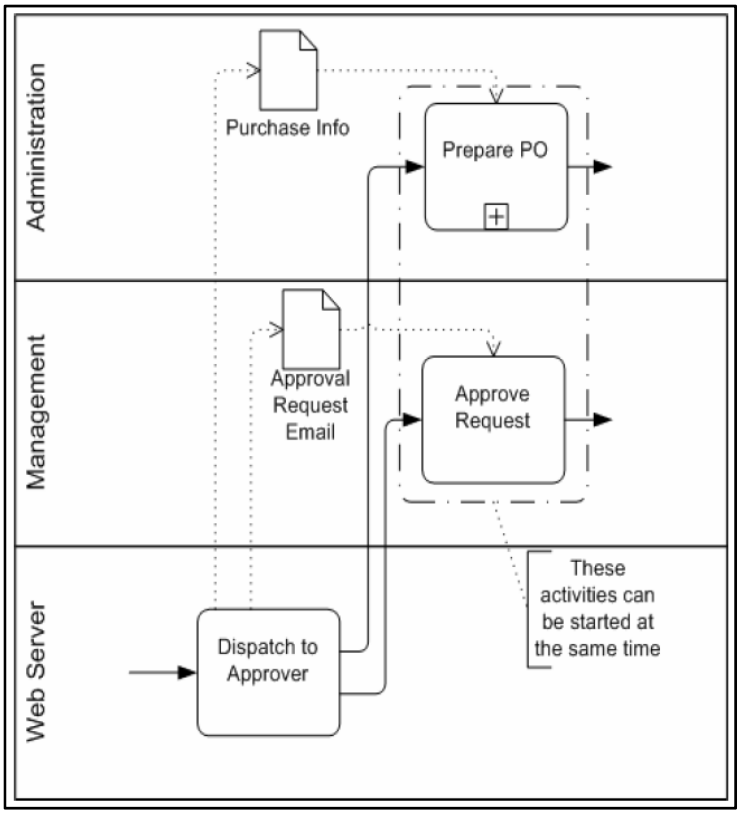
- ***Annotation*** – Le note servono al modellatore per esplicitare meglio un concetto legato ad un elemento, incrementando la leggibilità del BPD.



I-13. Annotation

E' importante sottolineare che la presenza di ***Artifacts*** non modifica il significato del BPD, ma ne chiarifica il senso al lettore. In figura I-13 possiamo vedere lo stesso diagramma presentato in precedenza arricchito di alcuni dettagli, che aiutano a comprendere e meglio dominare il processo. Ad esempio possiamo notare che l'inoltro della richiesta d'acquisto dal Web Server all' amministrazione avviene presentando un documento con le informazioni d'acquisto (le frecce delle ***Association Flow*** indicano che il documento è un output per il ***Task "***Dispatch to Approver" e un input per "Prepare PO"). Possiamo anche notare come le

attività del management e dell'amministrazione siano raggruppate e associate alla ***Annotation***, che indica la possibilità della loro esecuzione in parallelo.

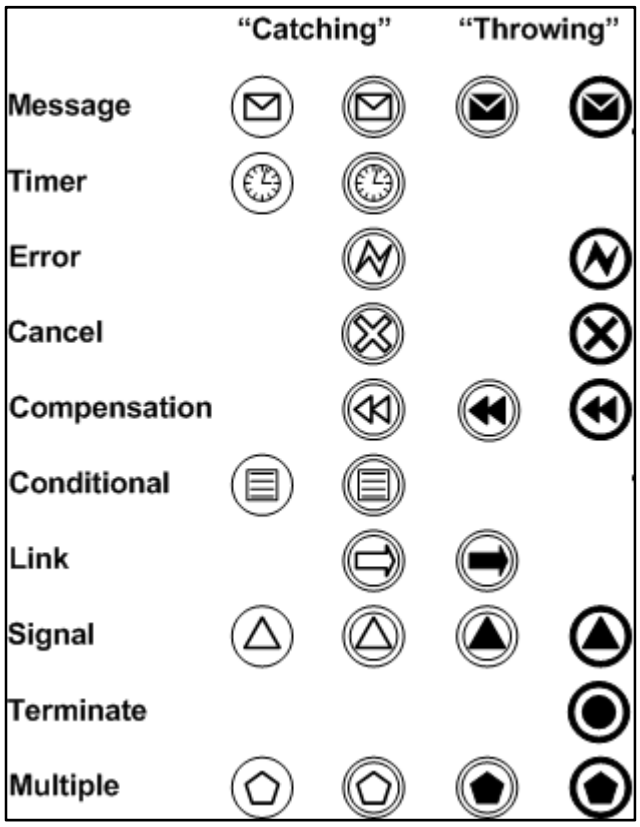


I-14. BPD con Artifacts

Capitolo 3. Set esteso degli elementi del BPD

Per poter meglio esprimere la complessità della natura dei processi di business che si vogliono modellare, lo standard (2)prevede un'estensione dell'insieme di elementi di base discussi nel capitolo precedente. Come vedremo vengono aggiunte alcune caratteristiche che meglio definiscono la natura di questi elementi.

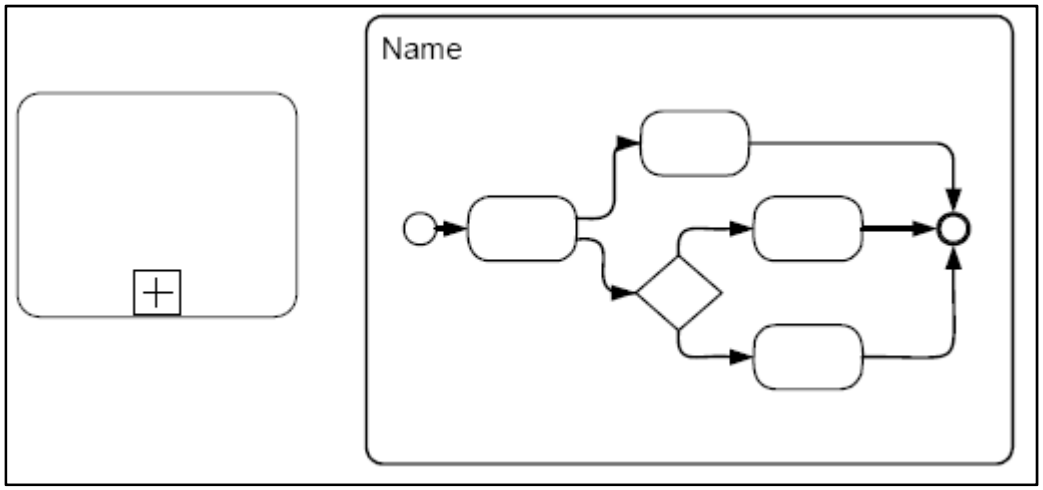
- ***Event Flow Object*** **–** Gli eventi vengono caratterizzati da dei trigger (per quanto riguarda ***Start*** e ***Intermediate Event***) che ne costituiscono la causa o risultati (nel caso di ***End*** e ***Intermediate Event***). Nella notazione differenzia i secondi dai primi utilizzando un bordo più grosso. In figura I-15 viene riportato l'insieme degli ***Event Flow Object*** previsti.



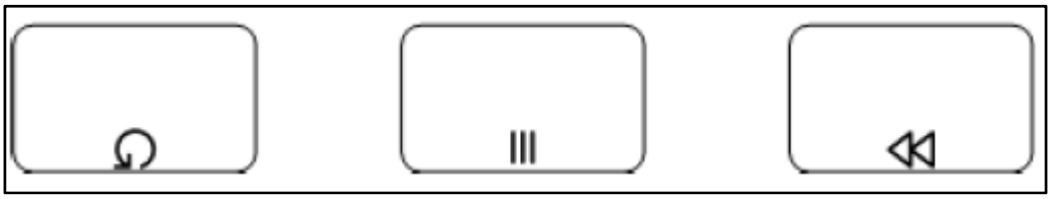
I-15. Set esteso degli eventi

- ***Activity Flow Object*** **–** Per identificare sottoprocessi non atomici è possibile riportare nel BPD stesso gli elementi che lo compongono oppure identificarli con un simbolo "+" in fondo ad essi(figura I-16). Vi sono poi tre simboli (figura I-17) che possono essere associati ai task per rappresentare cicli (***Loop Marker***), istanze multiple (***Multiple Instance Marker***), e task di compensazione (***Compensation Markers***). I task di compensazione (tipicamente azioni di ripristino di uno stato precedente) vengono considerati circostanze speciali, di conseguenza modellati in modo da evidenziarli rispetto al normale flusso modellato (figura I-18). Viene poi introdotto il concetto di processo ***ad - hoc***. In questo tipo di processo, la sequenzialità delle attività e il relativo numero delle esecuzioni necessarie non possono essere determinati a priori, ma vengono di volta in volta decisi dai partecipanti al processo. Ovviamente si tratta di un

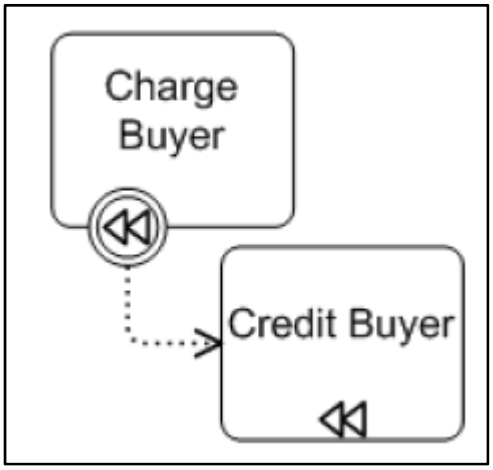
tipo particolare di processo, che quindi deve essere opportunamente identificato e gestito (figura I-19).



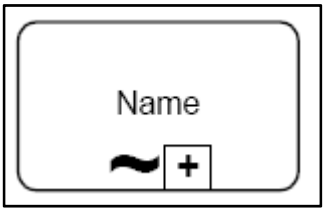
I-16. Collapsed Sub-Process e Expanded Sub Process



I-17. Loop Marker, Multiple Instance Marker e Compensation Marker

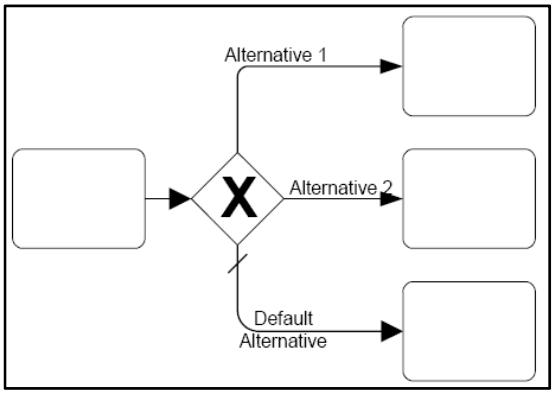


I-18. Compensation Task

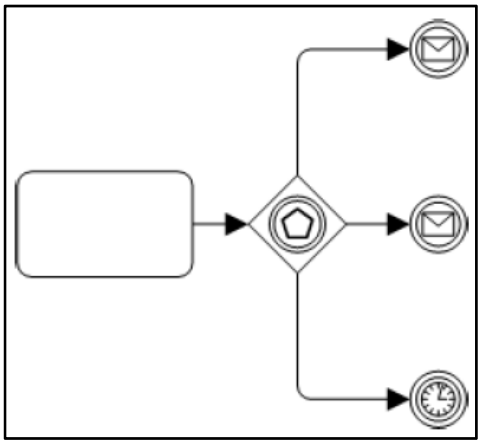


I-19. Ad - hoc Process

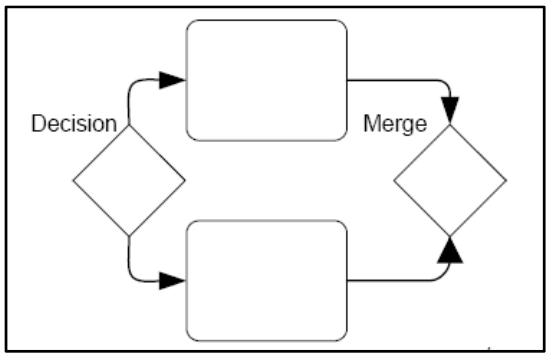
- ***Gateway Flow Object*** – I ***Gateway*** vengono caratterizzati dal tipo di comportamento che li caratterizza. Sono previsti
    - ***Exclusive Gateway*** – Se si vuole modellare una fork la cui decisione del percorso da seguire nell'esecuzione preveda uno solo dei cammini alternativi. Tale decisione può dipendere o dal risultato di un'espressione booleana basata sul valore di alcuni dati (***Data-Based***) o sul verificarsi di un evento (***Event-Based***). Questi ***Gateway*** vengono utilizzati anche per operazioni di merge, ma solitamente nel caso in cui la funzione sia più di connessione che non logica.



I-20. Data-Based Exclusive Gateway fork

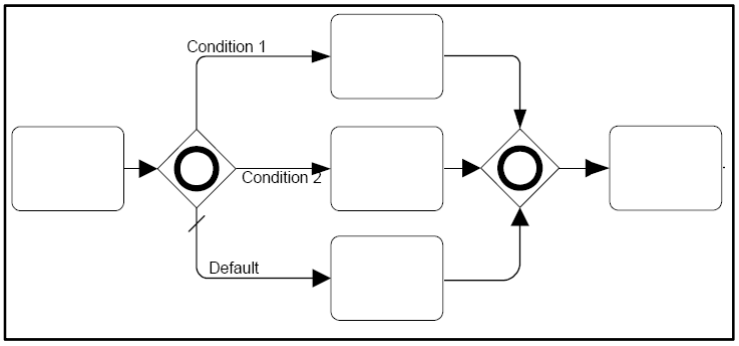


I-21. Event-Based Exclusive Gateway fork



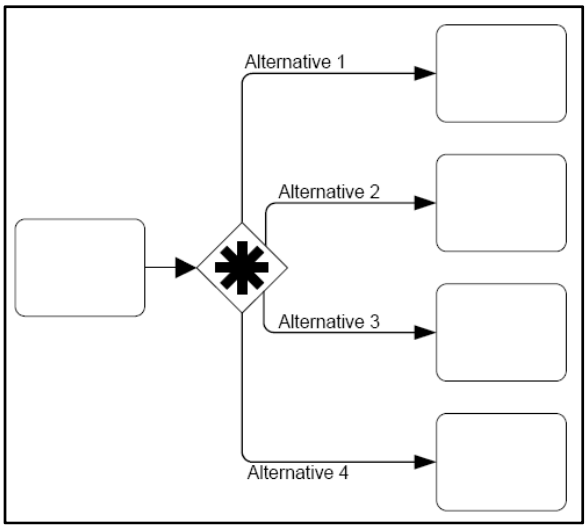
I-22. Exclusive Gateway Merge

- ***Inclusive Gateway*** - Per quanto riguarda la fork i percorsi alternativi sono caratterizzati da condizioni indipendenti l'una dalle altre, quindi più di un percorso può essere attivato. Nel caso di una merge, avviene una sincronizzazione dei percorsi in ingresso che risultino essere attivi.



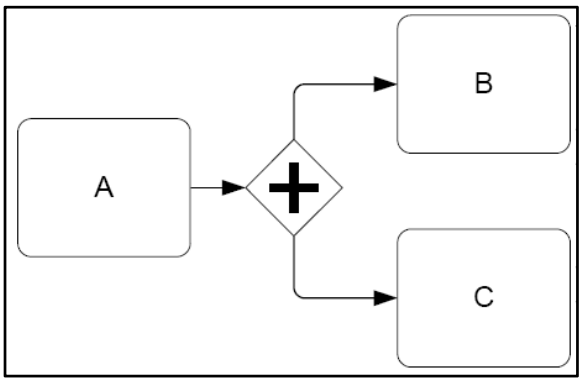
I-23. Inclusive Gateway fork / merge

- ***Complex Gateway*** – Utilizzato sia per merge che per fork il cui controllo non si modelli facilmente con gli altri tipi di ***Gateway***.



I-24. Complex Gateway

- ***Parallel Gateway*** – Utilizzato per creare e sincronizzare flussi paralleli. Non sarebbe necessario utilizzarli per la creazione ma se ne consiglia l'uso per una maggior chiarezza.



I-25. Parallel Gateway

- ***Sequence Flow*** – Qualora un percorso di esecuzione venga intrapreso sotto una particolare condizione che debba essere controllata, come ad esempio dopo un ***Gateway***, questi si identifica con un rombo all'inizio del percorso condizionale. Nel caso particolare di un percorso di default al posto del rombo troviamo un backslash.

I-26. Conditional e Default Sequence Flow

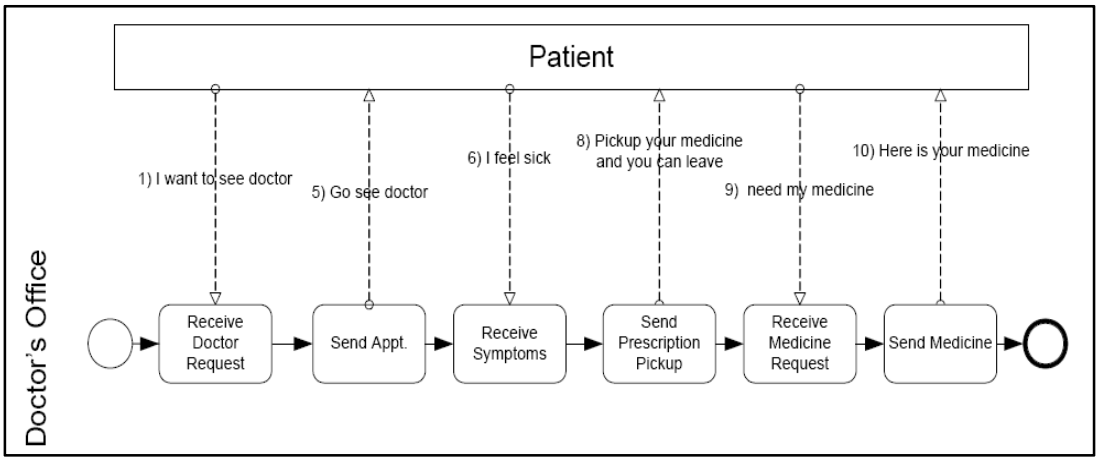
Capitolo 4. Utilizzi previsti per il BPMN

La modellazione dei processi ha lo scopo di comunicare una grossa quantità di informazioni per diversi scopi ed eterogenee situazioni. Il BPMN (1)è stato concepito per ricoprire la maggior parte di queste possibilità a differenti livelli di dettaglio. Lo standard del OMG (4) prevede in particolare tre tipologie generali di utilizzo.

- ***Private business processes*** **-** Appartengono a questa categoria i processi definiti interni ad una società (talvolta indicati col termine workflow o BPM processes). Scopo di questo tipo di modello è definire le attività dei vari processi e sottoprocessi sotto l'ottica della compagnia stessa.

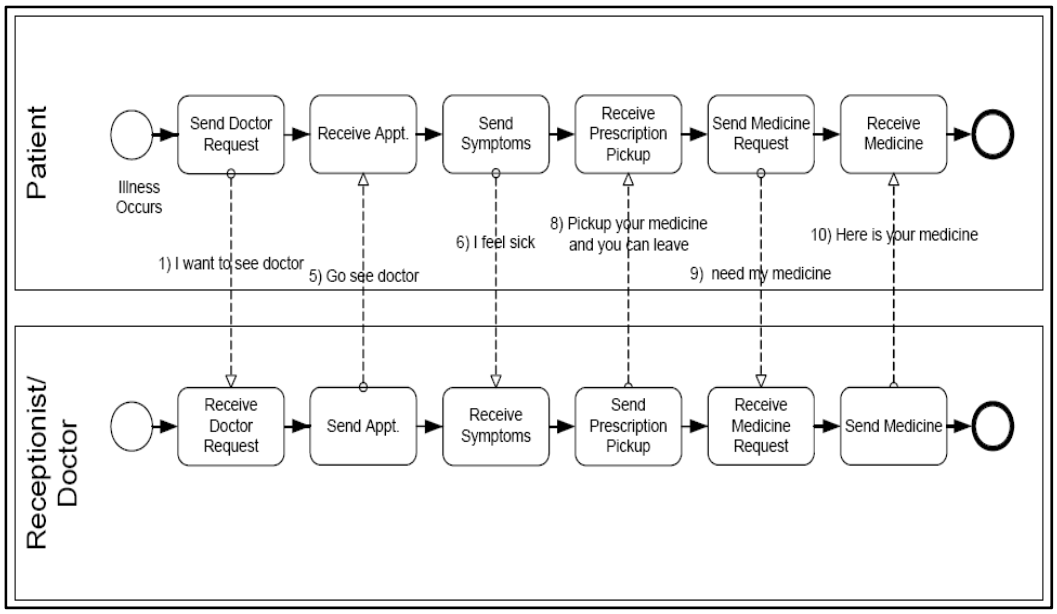
I-27. Private business process

- ***Abstract processes*** **–** In questo tipo di modello viene rappresentata l'iterazione tra un processo interno e un altro processo o partecipante ad esso. Mentre non vengono mostrate le attività interne al processo interno, il focus del diagramma è incentrato sulla comunicazione necessaria per il corretto interfacciamento ad esso.



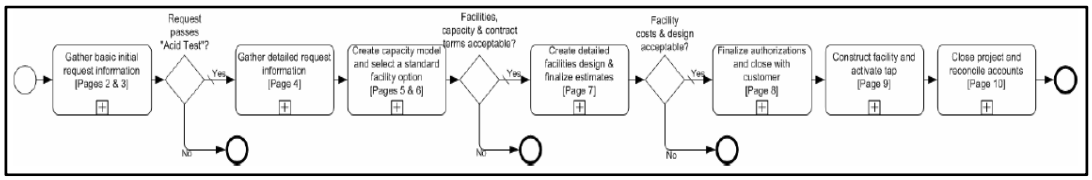
I-28. Abstract process

- ***Collaboration processes*** **–** Quest'ultimo tipo di modello, chiamato anche globale, prevede rappresenta l'interazione tra più entità del business, incentrando il focus sulla sincronizzazione e comunicazione tra le varie attività interne ad esse.

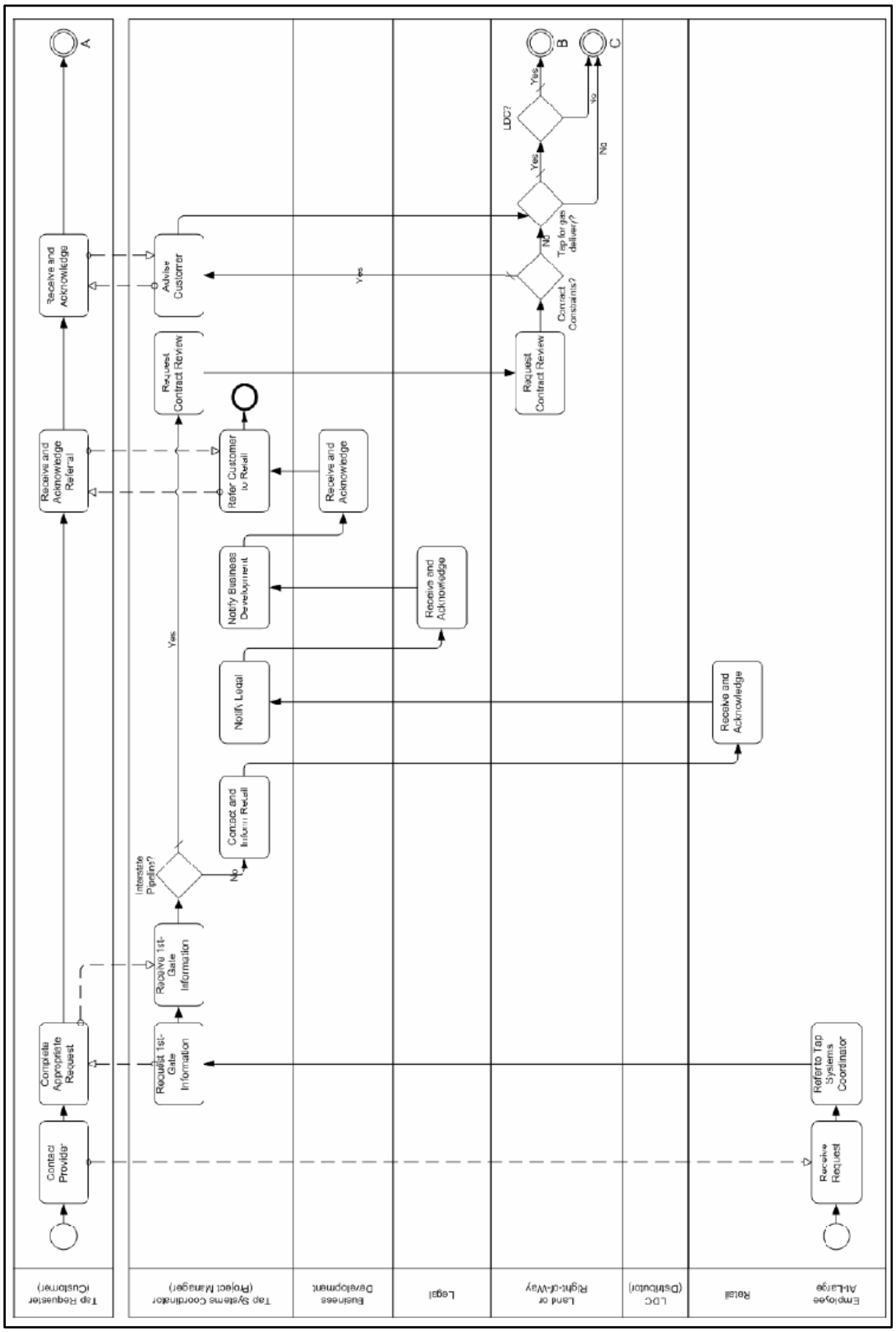


I-29. Collaboration Process

Come ben noto, il potere dell'informazione è legato alla capacità di sintesi di chi fornisce tali informazioni. Il BPMN (2) prevede la possibilità di modellare a diversi livelli di dettaglio i processi a seconda del tipo di utilizzo (e quindi utilizzatore) previsto. Un generico processo modellato come quello in figura I-30 potrà essere utile al top management, ma ai livelli + bassi della gerarchia sarà necessario un maggior livello di dettaglio (figura I-31) per poter effettivamente dominare il processo.



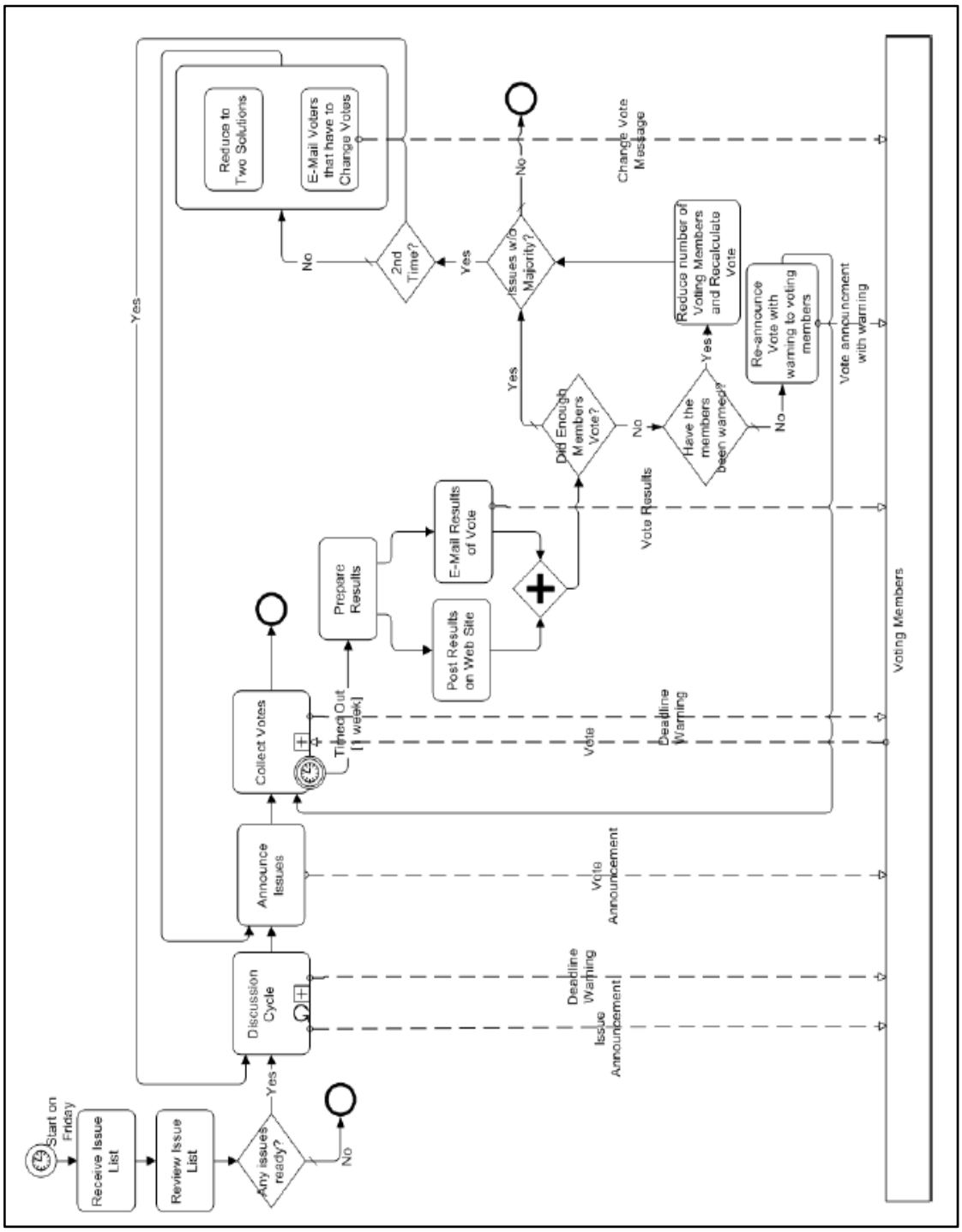
I-30. BPD ad alto livello



I-31. BPD nel dettaglio

Capitolo 5. Esempio di BPD

Nella specifica standard OMG (4) datata 17 gennaio 2008 del BPMN (2), al capitolo 11 viene presentato un processo (e un relativo BPD) esemplificativo. Scopo di questo capitolo sarà introdurre e analizzare tale esempio in modo da illustrare meglio la notazione.



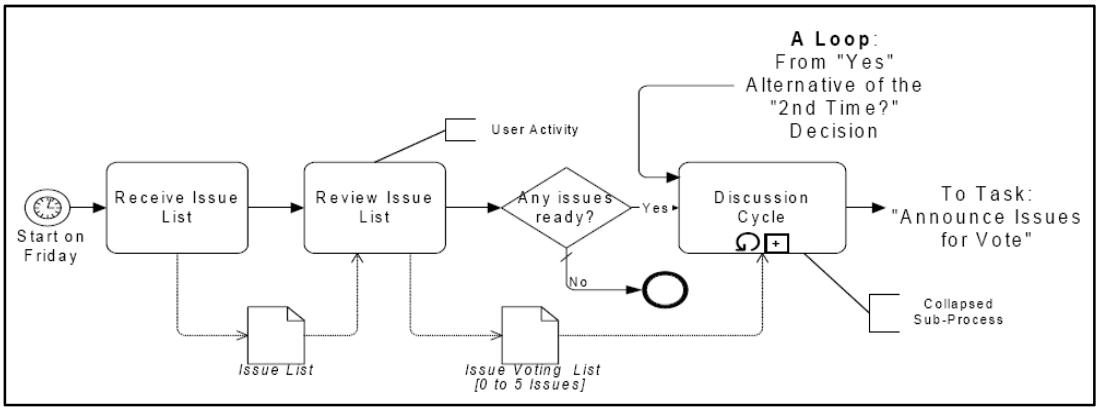
I-32. Processo d'esempio

Il modello descrive un processo per la risoluzione dei problemi attraverso tavole di discussione e votazione via e-mail, processo utilizzato dallo stesso OMG per la stesura delle

specifiche BPMN. Il processo è stato modellato secondo l'ottica del manager della lista dei problemi, di conseguenza (come possiamo notare in figura I-32. Processo d'esempio) gli altri membri votanti sono visti come figure esterne e la comunicazione avviene attraverso ***Message Flow***. Di seguito viene analizzato il processo suddividendolo per sezioni.

Fase iniziale del processo

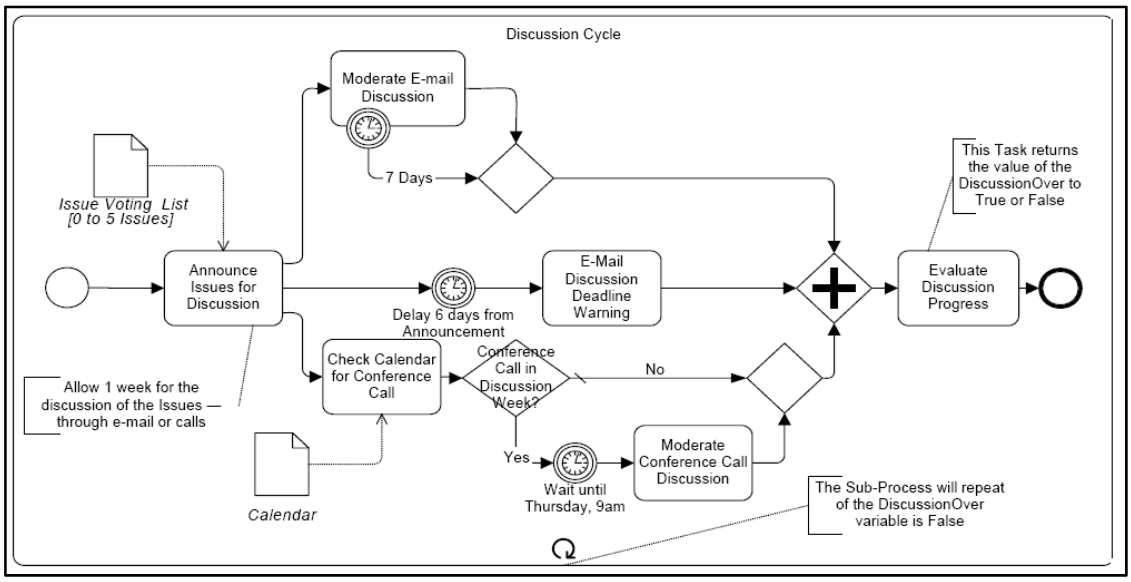
Possiamo vedere un ***Timer Start Event*** che indica il venerdì come inizio dell'intero processo. La prima attività è quella di ricevere la lista dei problemi, che viene poi passata al secondo ***Task*** dove viene effettuata un' analisi della stessa per decidere se e quali problemi sono pronti per essere passati al ***Sub-Process*** relativo al circolo di discussione dei problemi. Possiamo notare come il ***Task*** presenti la ***Annotation*** "user activity" ad indicare che l'attività sarà svolta dal manager della issue list. Il blocco decisionale attiva il ***Sub-Process*** se gli sono stati passati dei problemi.



I-33. Processo d'esempio - fase iniziale

Sub-Process Discussion Cycle

Entrando nel dettaglio in questo sottoprocesso, vediamo che il primo ***Task*** coinvolgerà l'issue list manager e, attivando le attività più a monte, verranno contattati gli altri partecipanti per attivare la discussione vera e propria (come possiamo vedere in figura I-32. Processo d'esempio questo corrisponde ad un ***Message Flow*** verso i partecipanti). Possiamo notare come vi siano 3 cammini alternativi paralleli sincronizzati poi con il ***Parallel Gateway***.



I-34. Sub-Process Discussion Cycle

Il primo cammino è caratterizzato da un ***Task*** di lungo termine "Moderate e-mail discussion" che presenta un ***Timer Intermediate Event*** al bordo. Quest'ultimo permette di fare avanzare il processo qualora il ***Task*** non finisca in tempo. Possiamo vedere che i ***Sequence Flow*** in uscita dell'***Event*** e dal ***Task*** vengono uniti con un ***Exclusive Gateway***, in modo che solo il primo dei due che si attivi possa arrivare al ***Parallel Gateway.*** Se non fosse stato fatto questo merge la sincronizzazione con gli altri cammini doveva avvenire con entrambe le uscite, soluzione che presenterebbe ovvi problemi di blocco del processo.

Il cammino centrale presenta un ***Timer Intermediate Event*** che permette una sospensione del cammino per 6 giorni, passati i quali vengono inviate le mail ai partecipanti per avvisarli che sta scadendo il tempo per la discussione.

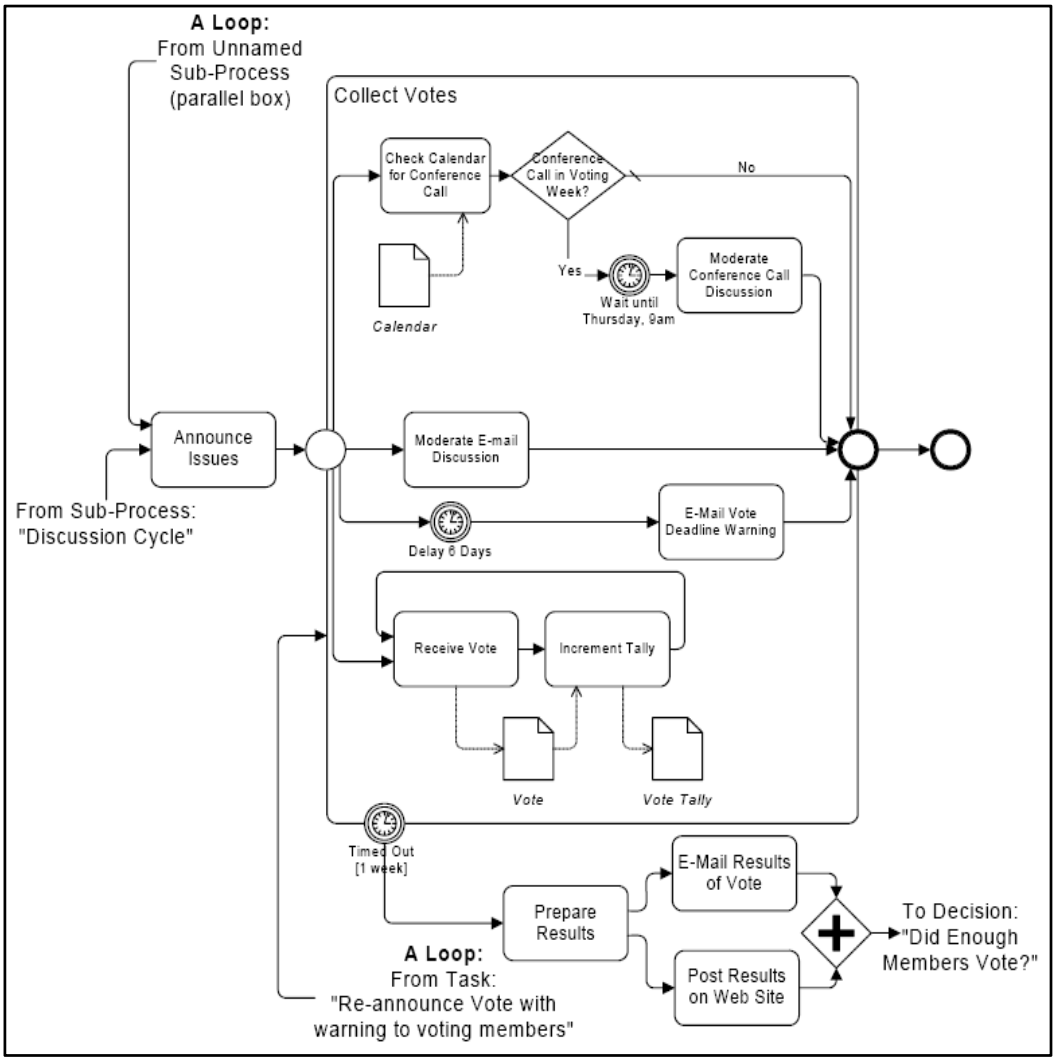
Il terzo cammino prevede invece la consultazione del calendario per la gestione di un'eventuale riunione di discussione. Possiamo notare il ***Timer Intermediate Event*** che, diversamente dagli altri, blocca il percorso non per un intervallo di tempo ma fino al verificarsi di un determinato istante nel tempo (Giovedì 9 Am è il momento in cui il gruppo di discussione dell'OMG si riuniva).

I tre cammini vengono sincronizzati con il ***Parallel Gateway*** che attiva il ***Task*** per il controllo della variabile "DiscussionOver", dal cui valore dipende la ripetizione del ***Sub-Process*** appena analizzato.

Sub-Process Collect Votes

Andiamo ora ad analizzare il ***Sub-Process "***Collect Votes". Come nel caso precedente, abbiamo una fork che attiva dei percorsi paralleli che portano alla conclusione dell'intero processo, mentre il sottoprocesso si interrompe dopo una settimana attraverso un ***Timer Intermediate Event*** del tutto simile a quello del ***Task*** "Moderate e-mail discussion" del caso precedente.

I primi tre cammini del sottoprocesso sono simili a quelli già affrontati nel "Discussion Cycle" relativi alla gestione delle riunioni, discussioni via mail e la notifica della deadline.



I-35. Sub-Process Collect Votes

Nel quarto ramo della fork troviamo un cammino particolare: un ciclo infinto. Questo è dovuto al fatto che non vi sono condizioni di uscita dal ciclo, quindi in teoria esso non termina. Questa progettazione è giustificata da due aspetti:

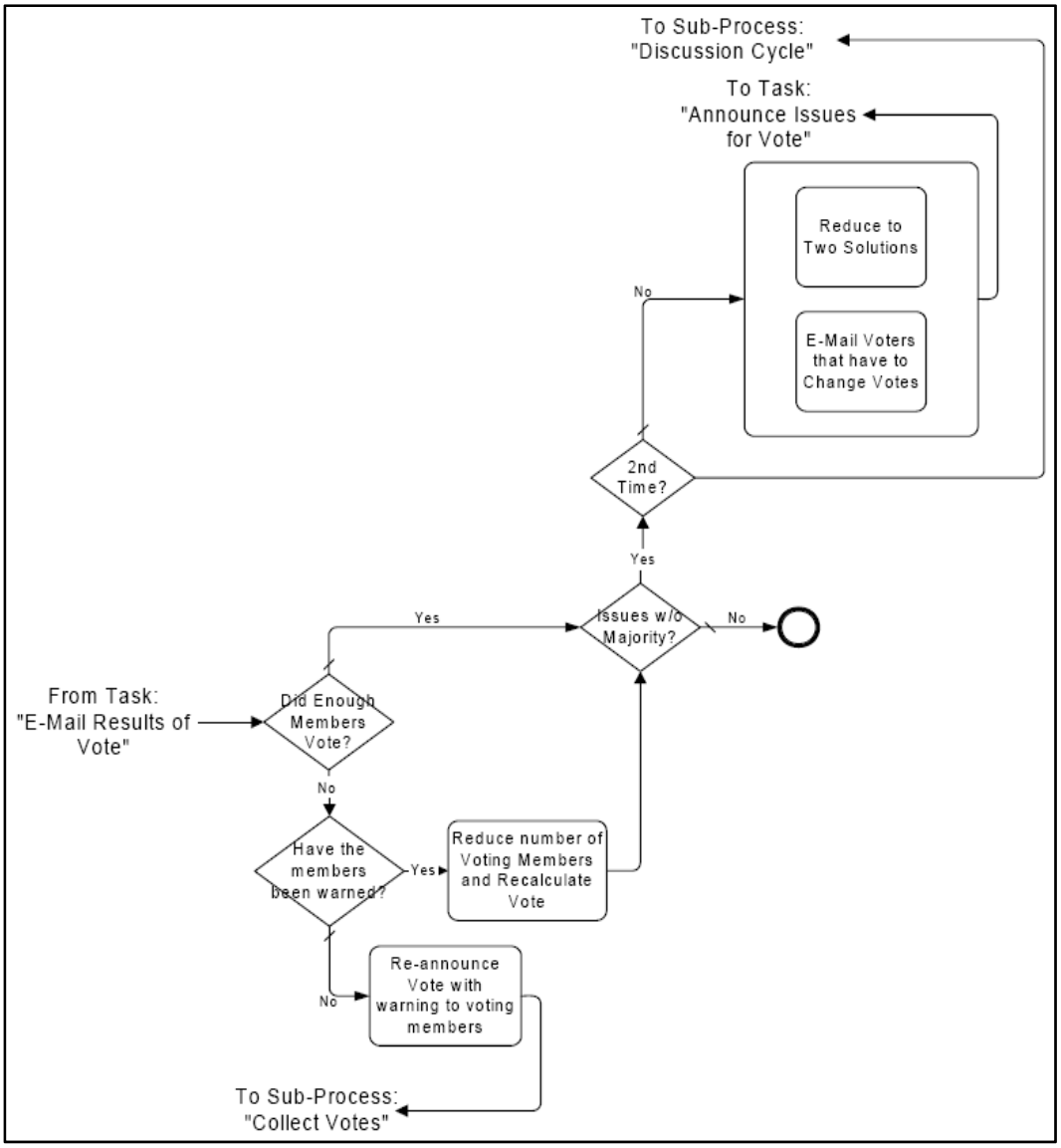
- Le attività del loop (votare e conteggiare i voti) sono da ritenersi infinte all'interno del sottoprocesso perché la politica del gruppo è che si possa votare un numero imprecisato di volte
- Esiste una reale anche se non ben identificata condizione di uscita del loop, ed è la terminazione (attraverso l'attivazione dell'***Timer Intermediate Event***) del sottoprocesso.

Scaduto il timeout relativo a "Collect Votes" vengono preparati i risultati e si passa alla fase finale quando sono terminati entrambi i ***Task*** "e-mail results of vote" e "post results on web site" (si ricordi la funzione di sincronizzazione del ***Parallel Gateway***).

Conclusione del processo

Nella parte conclusiva del processo vengono effettuate una serie di decisioni in cascata che applicano la politica decisionale generale del processo. Nello specifico :

- Il primo ***gateway*** controlla se almeno due terzi dei partecipanti debbano aver votato per far si che il problema in questione sia approvato
- Il ***gateway*** "Have the members been warned" applica la politica secondo un partecipante perde il diritto al voto se perde due votazioni successive. Se i partecipanti che non hanno votato devono essere avvisati si riprende con il sottoprocesso "Collect votes", in caso contrario si riduce il numero dei votanti e si ricalcola il voto.
- I rimanenti ***gateway*** servono per determinare se il processo debba finire o se si debba rivotare (sono previsti due giri di voti).



I-36. Processo d'esempio - fase finale

Prima di votare una seconda volta, vediamo modellato un sottoprocesso senza nome contenente due ***Task***. Il sottoprocesso in questo caso è utilizzato per creare un semplice percorso parallelo tra più attività.

Capitolo 6. Introduzione al linguaggio BPEL

Partendo dalla necessità di standardizzare i linguaggi di programmazione di alto livello per l'interazione dei Web Services (8), IBM (9), Microsoft (10), BEA Systems (11), SAP (12) e Siebel Systems (13) proposero la standardizzazione del BPEL4WS 1.1 (6) all' OASIS (Organization for the Advancement of Structured Information Standards) (14), che dall'introduzione della versione 2.0, avvenuta il 14 settembre 2004, viene conosciuto come WS-BPEL 2.0 o in maniera più generica BPEL (5) (15).

Il BPEL (5) è un linguaggio di orchestrazione, il cui scopo è quindi quello di descrivere la coordinazione tra più Web Services (8). La caratteristica di questi linguaggi è quella di definire un processo eseguibile che implichi scambio di messaggi tra questi nodi, in modo da essere controllato da uno di questi (orchestratore). Nel campo del BPM troviamo anche linguaggi chiamati di coreografia, il cui scopo è definire il tipo di interazione (protocolli) punto-punto tra i vari nodi coinvolti nell'esecuzione del processo, lasciando quindi un controllo decentrato ad ogni nodo. Mentre questi ultimi non sono eseguibili, i linguaggi di orchestrazione possono e vengono eseguiti da dei motori che li interpretano. Un documento BPEL si presenta infatti come un documento XML (16) con tutti i vantaggi che questo linguaggio porta.

L'applicazione del BPEL (5) può essere fatta per due tipologie di processo:

- ***Abstract Process*** – Si identificano di questo tipo quei processi che non debbano essere eseguiti. Questa caratteristica infatti permette di trascurare alcuni dettagli esecutivi, pur descrivendo in modo strutturato il processo.
- ***Executional Process*** - Questa categoria racchiude tutti i processi eseguibili. Per essere tali i processi vengono quindi definiti in completo dettaglio, in modo che il motore che li interpreta abbia le informazioni necessarie alla corretta esecuzione del processo descritto.

I principali obiettivi nella definizione del BPEL (5) sono stati:

- Definire processi di business che interoperino con entità esterne attraverso Web Services
- Definire processi di business utilizzando un linguaggio basato su XML.
- Definire un insieme di concetti di orchestrazione dei Web Services che possano essere utilizzati sia nei nella modellazione a uso esterno (***Abstract***) che quella ad uso interno (***Executable***).
- Fornire funzioni per la manipolazione dei dati.
- Supportare l'identificazione dei processi e delle istanze dei processi
- Supportare creazione e terminazione implicite del processo come ciclo di vita di base.
- Definire un modello transazionale basato su tecniche comprovate come azioni di compensazione, supporto agli errori etc.
- Utilizzare i Web Services come modello per il break-down dei processi
- Basarsi il più possibile su standard (proposti e approvati) relativi ai Web Services (8) (e.g. WSDL 1.1 (17)).

Andiamo ora a presentare brevemente gli elementi che compongono un documento BPEL (5).

- ***<process>*** - Elemento root del documento XML (16).
- ***<partnerLink>*** - Definizione dei partecipanti al processo. Vengono dichiarati e definiti i servizi e le funzionalità cui ci si interfaccia il Web Service (8) orchestratore. Ogni collegamento viene caratterizzato da ***<partnerLinkType>***, dichiarazioni WSDL (17) dei servizi partner e relativi role name utilizzati.
- ***<variable>*** - Dato utilizzato all'interno del processo, tipizzato in termini di WSDL message type, XML Schema type o XML Schema element (18).
- ***<receive>*** - Attività che permette la ricezione (bloccante) di un messaggio relativo ad un determinato ***partner Link.***
- ***<reply>*** - Istruzione di inoltro messaggio di risposta in relazione ad un messaggio ricevuto o ad un evento.
- ***<invoke>*** - Invoca un Web Service (8) partner, indicando se bisogna effettuare una serie di messaggio – risposta (comunicazione sincrona) che semplicemente l'invio di un messaggio (comunicazione asincrona).
- ***<assign>*** - Copia determinati dati in una variabile precedentemente dichiarata. E' possibile dichiarare se tali dati debbano prima essere validati secondo la loro definizione (XML (16) o WSDL (17)).
- ***<wait>*** - Utilizzato per sospendere l'attività per un determinato periodo di tempo o fino ad un determinato momento.
- ***<empty>*** - Attività di "no-op" (nessuna operazione" all'interno del processo di business. Utilizzato solitamente per sincronizzare attività concorrenti.
- ***<sequence>*** - Gruppo di attività da eseguirsi in ordine di definizione.
- ***<flow>*** - Gruppo di attività da eseguirsi con correntemente.

- ***<if>*** - Scelta esclusiva basata su una determinata condizione.
- ***<while>*** - Elemento utilizzato per generare loop. L'attività figlia viene eseguita e ripetuta finché non si verifica una determinata condizione.
- ***<repeatUntil>*** - Simile al ***while***, ma l'attività figlia viene eseguita almeno una volta.
- ***<forEach>*** - Simile al while, ripete l'attività figlia un numero preciso di volte.
- ***<pick>*** - Attività usata per recuperare uno dei possibili messaggi entro un determinato timeout. Conseguentemente viene eseguita l'attività figlia.
- ***<scope>*** - Elemento utilizzato per definire un'attività annidata con attributi (***partnerLink, variable, messageExchange***…) propri.
- ***<throw>*** - Lancia un'eccezione.
- ***<faultHanlders>*** - Definizione dei gestori delle eccezioni (***catch*** e ***catchAll***).
- ***<compensate>*** - Attività di compensazione di tutti gli ***scope*** interni al processo.
- ***<compensateScope>*** - Attività di compensazione di un determinato ***scope***.
- ***<correlations***> - Associa messaggi a specifiche istanze del processo.
- ***<exit>*** **-** Terminazione immediata dell'istanza del processo.

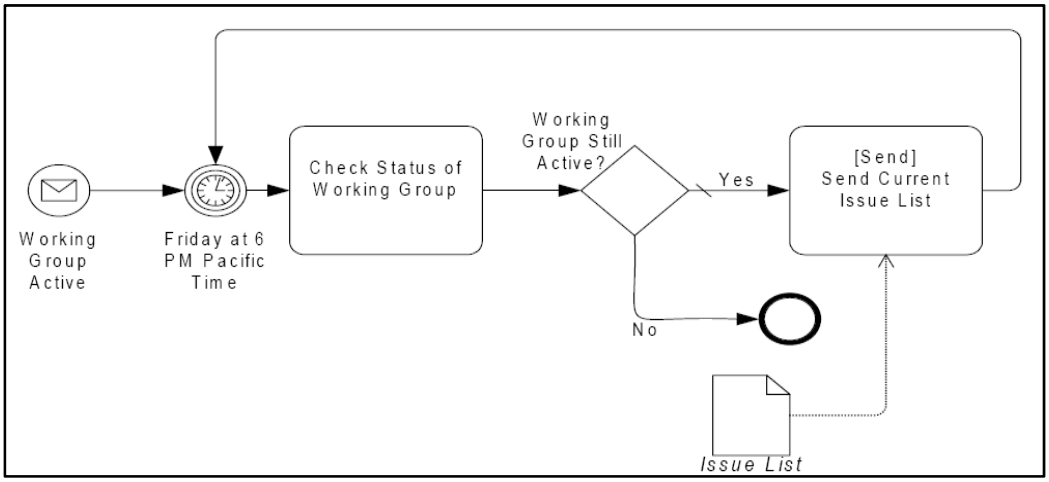
Capitolo 7. Passaggio da BPMN a BPEL

Nelle specifiche del OMG (4) del 17 gennaio 2008 del BPMN (2) viene illustrata una possibile mappatura della notazione nel BPEL. Come riportato in nota, nell' annesso A di tale documento, tale mappatura non è definitiva in quanto presenta ancora qualche problema di allineamento, ma verrà sistemata nella prossima pubblicazione. In attesa di uno standard completo, è comunque possibile nella maggioranza dei casi effettuare tale traduzione. Nello stesso standard viene riportato un esempio utile alla comprensione di questo procedimento, e in questo capitolo ne verrà effettuata una breve analisi.

L'esempio si basa sul processo presentato nel Capitolo 5 di questa sezione.

Fase iniziale del processo

Il primo problema da affrontare è lo start event del processo. In BPEL (5) i processi si attivano con una ***receive***, di conseguenza bisogna modellare un altro processo di servizio che ogni venerdì invii un messaggio al processo principale per poter farlo partire.



I-37. Processo starter di servizio

Il task "receive issue list" verrà quindi sostituito da una receive che creerà un'istanza del processo (attributo ***createInstance***), mentre l'intero processo viene descritto all'interno di un ***sequence***, in quanto vogliamo che le seguenti attività vengano svolte in cascata. Le proprietà utilizzate dal modellatore del processo per gestire le decisioni vengono mappate in BPEL in elementi ***variables*** chiamati "processData" .

Il task "review issue list" che come abbiamo visto viene svolta dall'issue list manager, viene mappata in un'istruzione sincrona ***invoke***, con inclusa una ***outputVariable***.

L'exclusive gateway "Any issues ready" viene mappato in uno ***switch***, e l'alternativa "no" (essendo progettata come default) viene mappata con l'***otherwise case*** dello stesso. In questo caso si vuole terminare il processo, quindi viene inserito solo il tag ***empty***.

Come possiamo notare in figura I-32, il processo presenta dei loop che coinvolgono alcuni sottoprocessi. Questi loop in BPEL verranno mappati con delle ***invoke*** a dei servizi chiamati " [(target of loop) activityName]_Derived_Process" (e.g. "Discussion_Cycle_Derived_Process"), che verranno chiamati dalle attività che in tali loop li precedono. Questa gestione dei sottoprocessi genera una serie di ***process*** e ***invoke*** che devono essere indirizzati. A questo scopo vengono definiti in WSDL (17) tali processi con la notazione "call_<process name>". Al momento dell'invocazione vengono passate le variabili necessarie all'esecuzione del sottoprocesso, al termine del quale restituirà il controllo al processo principale grazie a dei tag ***reply***.

Il sottoprocesso "Discussion cycle" è di tipo loop. In questo caso il loop è di facile realizzazione (non essendoci altri loop interlacciati ad esso da altri sottoprocessi) quindi la condizione è facilmente implementabile con un ***while***.

Sub-Process Discussion Cycle

Il sottoprocesso contiene tre percorsi paralleli che vengono mappati da un ***flow***. Andiamo ad analizzare come vengono mappati i tre percorsi alternativi:

- Nel cammino superiore il timer intermediate event viene gestito come un ***faultHandler*** con un suo ***scope***. Viene aggiunto a questo scope un ***eventHandler*** che contiene un ***for*** che serve da delay, alla fine del quale viene generato un fault attraverso l'uso di ***throw***. La gestione del fault contiene solo un elemento ***empty*** perché alla fine del percorso non ci sono altre attività.
- Il cammino centrale può essere mappato in due modi. Nella soluzione presentata viene usato un tag ***sequence***, ma una possibile alternativa è quella di usare dei link. Questa bivalenza rappresenta un'incognita nella generazione di documenti BPEL (5) da BPMN (1). Il timer intermediate event viene mappato come una ***wait***, mentre il task viene mappato come un ***invoke*** asincrono (non è prevista risposta).
- Nel terzo cammino possiamo notare che il primo task viene mappato con un ***invoke*** sincrono con due source, le due condizioni del controllo decisionale (notare che la condizione di default viene messa come negazione della precedente in quanto si attiva su *true*). In questo caso quindi il controllo viene implementato con dei link, uno dei quali effettua il ***wait*** e la ***invoke*** (ramo "yes") e l'altro solo una ***empty*** (ramo "no").

Sub-Process Collect Votes

Il sottoprocesso "collect votes", così come il task "announce issues", fa parte dei loop che abbiamo analizzato nella fase iniziale del processo, quindi anche per esso verrà generato un processo derivato.

I percorsi paralleli interni al sottoprocesso sono simili a quelli discussi nel paragrafo precedente, fatta eccezione per l'ultimo che presenta un ciclo infinto. Questi si realizza con un ***while*** la cui condizione è 1=0 (qualsiasi condizione sempre falsa sarebbe adatta). L'uscita dal ciclo infinto (così come dal sottoprocesso) viene realizzata similmente per quanto visto

nel primo ramo parallelo del sottoprocesso “Discussion Cycle”. Viene quindi definito uno ***scope*** per la gestione del fault generato dal ***onAlarm***.

Conclusione del processo

Come illustrato precedentemente, il passaggio all’ultima parte del processo avviene tramite la generazione di un fault (timeout di “collect votes”), quindi andrà inserita nel ***faultHandler*** del relativo ***scope***. All’interno dell’handler vengono implementati i controlli decisionali e i task (***switch*** e ***invoke***) e le ***invoke*** ai processi derivati per chiudere i cicli ai sottoprocessi.

Un caso particolare però è introdotto dal gateway “issues w/o majority”, cui si può arrivare da due percorsi alternativi. Questo è dovuto alla sovrapposizione di due controlli condizionali, possibilità comune in una notazione a diagramma ma impossibile in un linguaggio strutturato a blocchi come il BPEL (5). Le alternative sono due:

- Duplicare il codice di controllo e prevedere le varie possibilità dei cammini.
- Separare gli elementi in un processo derivato richiamabile da una ***invoke***.

La seconda opzione è quella scelta dall’OMG (4) in questo esempio, che conclude l’analisi della mappatura del processo in BPEL (5).

Capitolo 8. Ambienti di sviluppo

Come spesso accade, nonostante lo standard sia piuttosto giovane il mercato è già pieno di tool che permettono la progettazione in BPMN (1). Data l'estensibilità ma anche la non completezza della notazione, assieme a questi tool sono comparsi numerosi dialetti derivanti dal linguaggio, che lo amplificano e completano ma ne fanno perdere i vantaggi in termini di standardizzazione. Per questa tesi abbiamo preso in considerazione i tool che attualmente sul mercato sono le proposte più significative:

- Intalio (19) Designer 5.2.1.158
- Oracle (20) BPM Studio 10.3.1.0.0
- E-Clarus Business process modeler 2.0
- ActiveVOS (21) Designer 6.0.2.0

Dovendo confrontare le potenzialità proposte dai due approcci (BPMN (1)+ BPEL (5) e YAWL (22)), abbiamo escluso E-Clarus (23) in quanto si tratta di uno strumento orientato più alla modellazione / trasformazione che non all'esecuzione dei processi.

Di seguito vengono confrontate le caratteristiche offerte dalle varie soluzioni commerciali, che sono state la chiave per la scelta dello strumento utilizzato nel corso di questa tesi.

Tabella 1. Disponibilità nei Sistemi Operativi

OS	Oracle (20) BPM Studio	ActiveVOS (21)	Intalio (19) Community	Intalio (19) Enterprise
Windows Vista (10)	D			
Windows XP (10)	D	D	D	D
Windows 2008 (10)	E			
Windows 2003 (10)	E	E	E	E
Windows 2000 (10)	E		D+E	D+E
Linux Red Hat 5 (24)	D+E	D+E	D+E	D+E
Linux Red Hat 4 (24)	D+E	D+E	D+E	D+E
Linux SUSE 10 (25)	D+E		D+E	D+E
Linux SUSE 9 (25)	D+E		D+E	D+E
Mac OS X (26)			D	D
OEL 4 (20)	E			
OEL 5 (20)	E			
IBM AIX 5.3 (9)	E			E
IBM AIX 5.2 (9)				E
Sun Solaris 10 (27)	E	E		E
Sun Solaris 9 (27)				E
Sun Solaris 8 (27)				E
HP-UX 11.3 (28)	E			E
HP-UX 11.2 (28)				E

Tabella 2. Application Server Supportati

Application Server	Oracle (20) BPM Studio	ActiveVOS (21)	Intalio (19) Community	Intalio (19) Enterprise
Tomcat 5 (29)	V	V		V
Jboss 4 (30)		V		V
Jboss Enterprise Application Framework 4 (30)		V		
Bea Weblogic 10 (11)	V			
Bea Weblogic 9.2 (11)		V		
IBM Websphere 6 (9)		V		V
Geronimo 2 (31)			V	V
Oracle Weblogic 10 (20)	V			

Tabella 3. Database Supportati

Database	Oracle (20) BPM Studio	ActiveVOS (21)	Intalio (19) Community	Intalio (19) Enterprise
MySQL 5 (32)		V	V	V
Oracle DB 10g (20)	V	V		V
Oracle DB 9i (20)	V			V
Oracle DB 8i (20)				V
MS SQL Server 2005 (10)	V	V		V
MS SQL Server 2000 (10)				V
PostGre SQL 8.2 (33)				V
PostGre SQL 8.1 (33)				V
Sybase ASE 15 (34)	V			V
Sybase ASE 12 (34)				V
IBM DB2 8 (9)	V	V		V
IBM DB2 9 (9)	V			V
Derby 10 (35)			V	V

Tabella 4. Funzionalità offerte

General Features	Oracle (20) BPM Studio	ActiveVOS (21)	Intalio (19)Community	Intalio (19) Enterprise
Open Source		V	V	V
Free Availability			V	
Clustering	V	V		V
failover	V	V		V
Distributed transactions	V			V
real-time dashboard	V	V		V
process driven user portal	V			V
AJAX IDE				V
Real run mode			V	V
Simulation mode	V	V		
Round - trip Engineering		V	V	V

Tabella 5. Supporto degli standard

	Standard Supportato	Oracle (20) BPM Studio	ActiveVOS (20)	Intalio (19)
Process	Aris EPC (36)	-		V
	Oasis (14) WS- BPEL 1.1 (15)	-	V	V
	Oasis (14) WS- BPEL 2.0 (5)	-	V	V
	Oasis (14) WS-Human Task (37)	-	V	
	Oasis (14) BPEL 4 People (38)	-	V	V
Interface	WSDL 1.1 (17)	-	V	V
	XML Schema 1.0 (18)	-	V	V
	XPATH (39)	-	V	V
	XQUERY (40)	-	V	V
	XFORMS (41)	-		V
	XSLT (42)	-	V	
Protocols	SOAP 1.1 (43) over HTTP (44)/HTTPS (45)	-	V	V
	SOAP 1.1 (43) /Plain XML (16) over JMS (46)	-	V	V
	REST (47)	-	V	V
	JAVA (48)	-	V	V
	WS-Addressing (49)	-		V
	WS-Reliable Messaging (50)	-	V	V
Security	WS-I Basic Profile (51)	-	V	V
	WS-Security (52)	-	V	V

	SAML 1.1 (53)	-	V	
	LDAP v3 (54)	-	V	V
Governance	WS-Policy (55)	-	V	
	UDDI v2 (56)	-	V	
	UDDI v3 (56)	-	V	
Discovery	WSIL (57)	-	V	
Management	WS-Distributed Management (58)	-	V	

Possiamo notare dalla Tabella 1 come i vari prodotti si possano utilizzare su diverse piattaforme. In particolare abbiamo distinto i casi in cui il sistema operativo supporti:

- L'ambiente di progettazione (D)
- L'ambiente di esecuzione (E)
- Entrambi (D+E)

In alcune tabelle abbiamo inoltre ritenuto opportuno differenziare Intalio (19) Designer 5.2.1.158 in due distribuzioni, quella community (disponibile gratuitamente ma con funzionalità limitate) e quella enterprise (completa ma a pagamento).

Dalla Tabella 4 possiamo notare che mentre le varie soluzioni si diversificano tra loro per supporto di diverse tecnologie in commercio (application server, sistema operativo e DB) le funzionalità generiche offerte sono molto simili tra loro.

Infine in Tabella 5 riportiamo quali standard vengono supportati dal sistema. Purtroppo in questo ultimo caso non sono stati riscontrati dati sufficienti per poter comparare correttamente anche la soluzione offerta da Oracle (20). Per quanto riguarda quest'ultima piattaforma, essa principalmente non utilizza l'approccio BPMN (1)+ BPEL (5), bensì traduce i BPD modellati in un linguaggio proprietario. Un'analisi approfondita di tale approccio esula dallo scopo di questa tesi, ma sarà un'interessante caso da prendere in esame assieme al confronto tra i due approcci BPMN (1) + BPEL (5) e YAWL (22).

La scelta su quale sistema utilizzare è stata guidata per lo più dal supporto degli standard, in quanto:

- si ritiene sia l'aspetto cardine dell'approccio BPMN (1) + BPEL (5)
- il supporto di sistemi operativi, application server e DB non influenzano lo scopo di questa tesi
- le funzionalità offerte dai vari sistemi sono comunque simili (Tabella 4)

La scelta è quindi ricaduta sul sistema offerto da ActiveVOS (21). Questi presenta inoltre un'interfaccia di sviluppo apparentemente più orientata al BPM e meno legata all'ambiente Eclipse (59) su cui si basano anche gli altri sistemi sopra citati, fatta esclusione per Oracle

(20) BPM Studio 10.3.1.0.0 che è stato però escluso per il suo approccio orientato maggiormente verso un linguaggio proprietario.

Sezione II. Approccio al BPM con YAWL

Capitolo 9. Introduzione a YAWL

Partendo da uno studio sulle potenzialità e differenze tra i vari sistemi di workflow management (60), nel 2002 la Eindhoven University of Technology (61) e la Queensland University of technology (62) hanno sviluppato un nuovo linguaggio basato sulle reti di Petri : YAWL (Yet Another Workflow Language) (63).

Lo studio si incentrava sull'analisi dell'applicabilità di 15 soluzioni commerciali (tra le più diffuse in quegli anni), a 20 design patterns identificati come maggiormente comprensivi della complessità di modellazione dei business processes. Questi pattern sono stati classificati in 6 categorie, che riportiamo di seguito:

- ***Basic control flow patterns*** - Costrutti basilari presenti in quasi tutti i sistemi analizzati per modellare flussi sequenziali, paralleli e condizionali.
- ***Advanced branching and synchronization patterns*** – Costrutti avanzati per gestire ramificazioni e sincronizzazione come ad esempio ***Synchronizing Merge***.
- ***Structural Patterns –*** Identificano punti di ingresso e di uscita da blocchi di costrutti.
- ***Patterns involving multiple instances –*** Patterns per la gestione di istanze multiple di parti di un singolo processo all'interno di una sua esecuzione.
- ***State - based patterns –*** Pattern orientati alla gestione dello stato del processo.
- ***Cancellation Patterns*** – Pattern per la cancellazione delle attività svolte in seguito al verificarsi di un particolare evento.

Per una presentazione più approfondita dei pattern utilizzati per l'analisi dei sistemi di workflow management si rimanda al Capitolo 11.

Nella maggioranza dei casi i vari linguaggi offrivano un supporto diretto all'implementazione di meno della metà di questi design pattern. Questo non vuol dire che non fosse possibile modellare tali realtà usufruendo di quei sistemi, ma la mancanza di tale supporto rendeva la modellazione oltremodo macchinosa, degradando drasticamente la manutenibilità dei modelli.

Alla luce di questo studio, è stata presa in considerazione l'ipotesi di utilizzare una notazione che si basasse sulle reti di Petri estese (***Coloured Petri Nets***) (64). Con reti di Petri estese si intendono delle notazioni derivanti dalle reti di Petri originarie (65) (inventate nell'agosto del 1939 da Carl Adam Petri) cui sono stati aggiunti concetti vari come l'identificazione dei token, temporizzazione etc. Molte di queste estensioni sono riconducibili alle reti di Petri (65) originali, garantendo quindi le stesse proprietà matematiche di analisi che contraddistinguono questa notazione. I motivi principali che hanno suggerito l'utilizzo di questo linguaggio sono:

- Semantica formale affiancata da una notazione grafica semplice
- Modellazione basata sullo stato invece che esclusivamente sugli eventi

- Abbondanza di tecniche di analisi dei modelli basate sui formalismi matematici del linguaggio

Le reti di Petri (65) però non bastavano (63)come notazione per la modellazione dei processi di business in quanto:

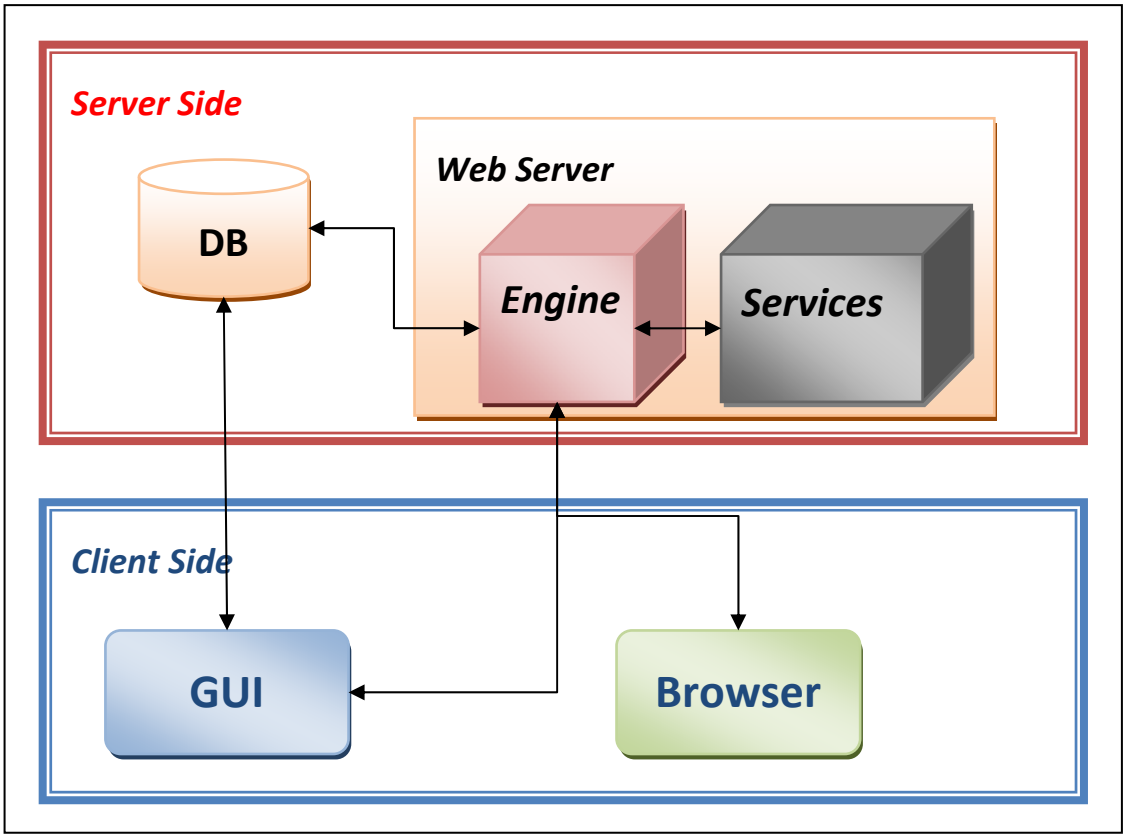
- Manca un supporto specifico per la modellazione dei pattern per la gestione delle istanze multiple
- Difficoltà di modellazione dei pattern per la sincronizzazione avanzata
- Le transizioni sono attivate sempre localmente, comportando difficoltà nella modellazione dei pattern per la cancellazione delle attività svolte.

YAWL come WFMS

YAWL (63) si presenta quindi come soluzione alternativa ai WFMS già presenti sul mercato. Il motore e l'interfaccia grafica sono scritti in Java (48) sotto licenza LGPL (66). L'interoperabilità con altri sistemi è garantita dall'utilizzo intensivo di XML Schema (18), XPath (39), XQuery (40) e XForms (41), ed è compatibile con SOAP (43) e WSDL (67). Con l'obiettivo di creare un WFMS

- Espressivo – attraverso il supporto dei workflow pattern
- Conciso – orientato alla soluzione piuttosto che alla tecnologia
- Flessibile – permettendo la creazione di webservices da integrare nel linguaggio
- Affidabile – attraverso i formalismi delle reti di Petri (65) è possibile analizzare e verificare un workflow (68)

YAWL (22)si presenta con un'architettura client-server come in figura (69) .



II-1. Architettura Client - Server di YAWL

La persistenza dei dati è inoltre garantita dall'interfacciamento con Postgre (33) DBMS, mentre l'accesso ai servizi avviene attraverso il Web Server Apache Tomcat (29).

Nel seguito verrà quindi esposto YAWL 2.0 (22) come strumento integrato per la modellazione dei processi di business, in modo da poterlo poi confrontare con l'utilizzo di BPMN (1) + BPEL (5). La presentazione si riferirà anche all'ambiente di esecuzione di YAWL 2.0 (22), in quanto essere parte integrante della soluzione prevista dalla YAWL foundation (22).

Capitolo 10. Notazione YAWL

Come già illustrato, YAWL (63) deriva dalle reti di Petri (65) e ne conserva la natura rigorosa di modello matematico. Infatti ogni elemento del linguaggio viene definito in termini di funzioni tra insiemi, permettendo un'analisi formale delle sue proprietà in termini di teoremi e relative dimostrazioni. Questa presentazione si limiterà alla presentazione del linguaggio come strumento per la mappatura dei processi, similmente a quanto fatto per il BPMN (1). In questo capitolo infatti verranno elencati gli elementi che compongono il linguaggio, mentre di seguito ne verrà illustrato l'utilizzo nei vari pattern necessari a modellare i processi di business.

- ***Condition*** – Rappresenta uno stato del processo.

II-2. Condition

- ***Input Condition*** – Stato iniziale del processo.

II-3. Input Condition

- ***Output Condition*** – Stato finale con cui si conclude il processo.

II-4. Output Condition

- ***Atomic Task*** – Rappresenta una singola attività che deve essere eseguita (non c'è distinzione tra attività umana e attività automatica).

II-5. Atomic Task

- ***AND - Split*** **–** Task al termine del quale vengono attivate tutte le connessioni in uscita.

II-6. AND - Split Task

- ***AND-Join –*** Task che si attiva quando sono stati attivati tutti i flussi di controllo in ingresso.

II-7. AND - Join Task

- ***OR - Split –*** Task al termine del quale vengono attivate solo alcune connessioni in uscita.

II-8. OR - Split Task

- ***OR-Join –*** Task che si attiva quando una o più connessione in ingresso si attiva. Dopodiché finché il task continua i flussi in ingresso non possono più attivarsi.

II-9. OR - Join Task

- ***XOR - Split -*** Task al termine del quale viene attivato solo un flusso in uscita.

II-10. XOR - Split Task

- ***XOR - Join –*** Task che si attiva ogni volta che un flusso in ingresso si attiva.

II-11. XOR - Join Task

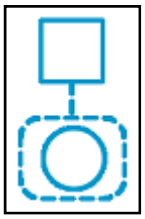
- ***Composite Task –*** Task composto, rappresenta un sottoprocesso del processo che si modella.

II-12. Composite Task

- ***Multiple Instance Task*** **–** Task che prevede l'esecuzione di istanze multiple concorrenti.

II-13. Multiple Instance Task

- ***Cancellation Region*** **–** Una volta attivato il Task associato alla zona tratteggiata, gli elementi in essa contenuti vengono disattivati.



II-14. Cancellation Region

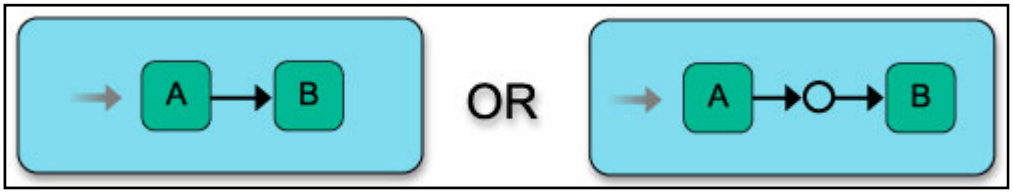
Prima di passare alla presentazione dei pattern bisogna concludere la descrizione della notazione descrivendo una specifica di workflow. In YAWL (63) una specifica di workflow viene definita come un insieme di reti estese di workflow (***EWF - Nets***) che formano una struttura ad albero, dove i ***Task*** possono essere atomici(foglie della struttura) o composti (nodi dell'albero). Ogni ***EWF - net*** è composta da ***Task*** e da ***condizioni***, in particolar modo finisce con un'unica ***input condition*** e un'unica ***output condition***.

Capitolo 11. Design Patterns in YAWL

Abbiamo introdotto nel Capitolo 9 come gli sviluppatori di YAWL (63) abbiano preso in esame 20 pattern utili alla mappatura dei processi business e li abbiano scelti come metro di valutazione per i vari linguaggi di workflow management. Si intende ora presentare nel dettaglio i vari pattern suddivisi nelle diverse categorie.

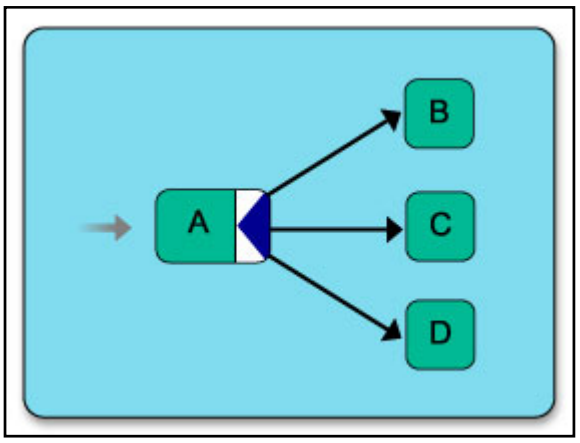
Basic Control Flow Patterns

1. ***Sequence*** – Rappresenta la semplice sequenzialità che esiste tra due diversi elementi di un modello.
   Contrariamente a ciò che accade nelle reti di Petri è possibile connettere oggetti di transizione (quali i ***Task***) senza utilizzare oggetti di stato (quali le ***conditions***).



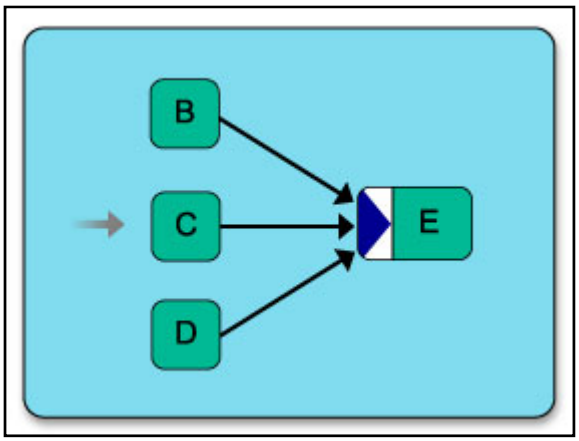
II-15. Pattern Sequence

2. ***Parallel Split –*** I ***Task*** B, C e D sono eseguiti in parallelo subito dopo il ***Task*** A. Il ***Task*** A si dice di tipo AND - Split.



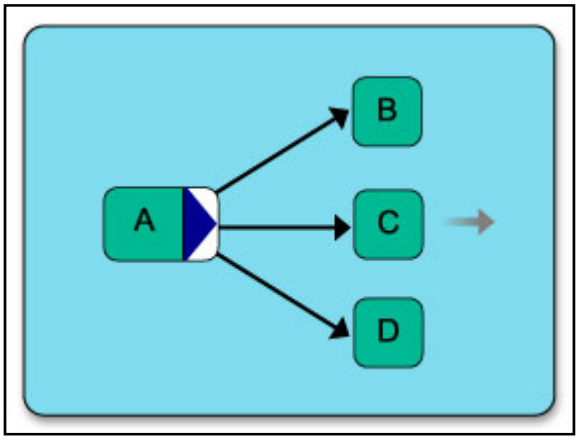
II-16. Pattern Parallel Split

3. ***Synchronization –*** Il ***Task*** E può essere eseguito solo dopo la conclusione di B, C e D. E implementa un ***Task*** di tipo AND-Join.



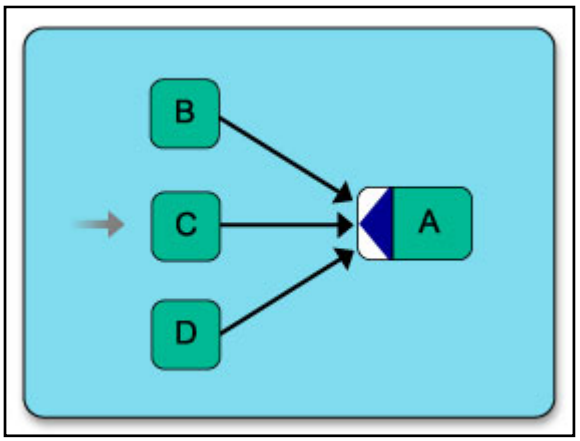
II-17. Pattern Synchronization

4. ***Exclusive Choice*** – Solo uno tra i ***Task*** B, C e D viene attivato al termine di A (XOR - Split). E' necessario indicare le condizioni di attivazione dei diversi cammini, attualmente YAWL (63) prevede l'utilizzo del linguaggio XPath (39).



II-18. Pattern Exclusive Choice

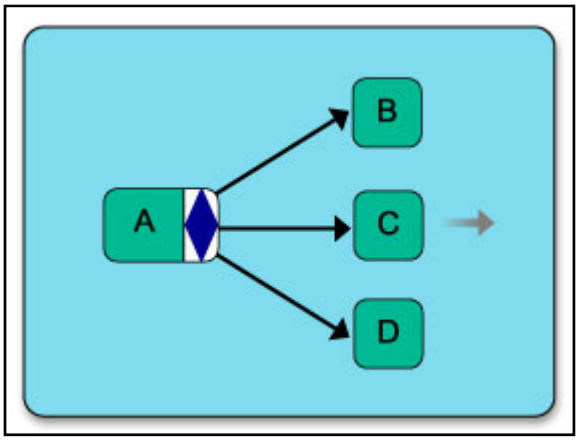
5. ***Simple Merge –*** A viene attivato quando uno o più tra i ***Task*** B, C o D si conclude (XOR - join). Questo pattern viene utilizzato quando non è necessario avere una sincronizzazione a monte di A.



II-19. Pattern Simple Merge

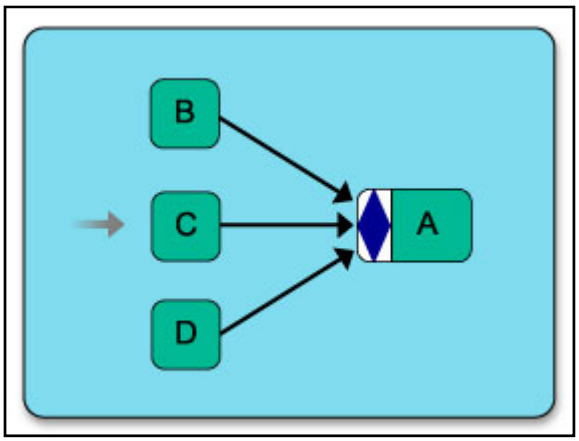
Advanced Branching and Synchronization Patterns

6. ***Multiple Choice –*** Similmente all'***exclusive choice***, l'attivazione dei vari ***Task*** a valle di A (OR - Split) dipende da dei predicati condizionali, ma questa volta l'attivazione di un dei ***Task*** tra B, C o D non preclude l'attivazione degli altri.



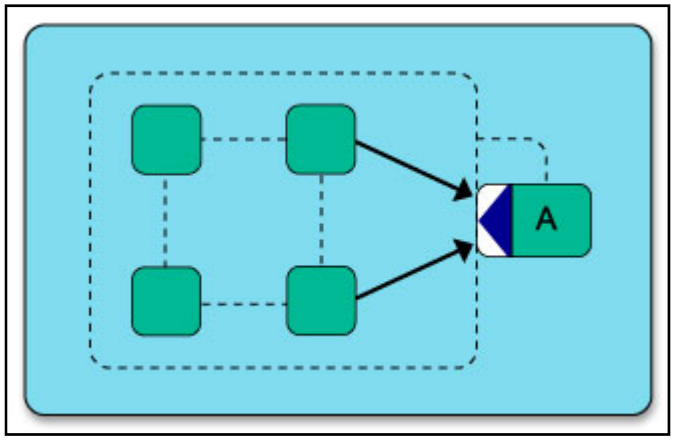
II-20. Pattern Multiple Choice

7. ***Synchronizing Merge –*** Il ***Task*** A (OR - join) si attiva quando almeno uno dei ***Task*** a monte si conclude, ma non può venire attivato nuovamente prima della sua conclusione.

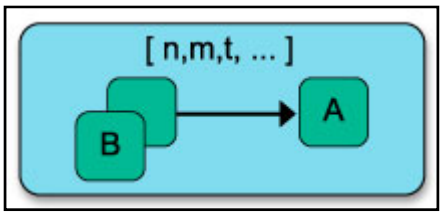


II-21. Pattern Synchronizing Merge

8. ***Multiple Merge*** **–** Questo pattern viene implementato come il pattern ***Simple merge***.
9. ***Discriminator*** **–** Il pattern prevede che un'attività attenda che almeno uno dei rami a valle si completi per attivarsi e poi ignori i successivi input attivi. L'attività si deve poi resettare quando tutti i rami sono stati attivati. Una situazione simile può verificarsi ad esempio nelle trasmissioni di rete, dove venga implementata una politica at most once. L'applicazione di questo pattern in YAWL (22) prevede inoltre la cancellazione delle altre attività a monte, visto che tanto non andranno ad influire sull'esecuzione dell'attività in considerazione. Vi sono due possibili soluzioni per modellare questo pattern. La prima di queste utilizza il pattern ***Cancel Case*** racchiudendo le attività che stanno a monte di A (figura II-22). La seconda soluzione utilizza delle proprietà aggiuntive dei ***Task.*** Ogni ***Task*** può infatti avere multiple istanze che vengono create al momento dell'attivazione del ***Task*** stesso. Il ***Task*** si conclude o quando tutte le sue istanze vengono completate oppure al raggiungimento di un numero di soglia (***threshold***) preventivamente definito (figura II-23).



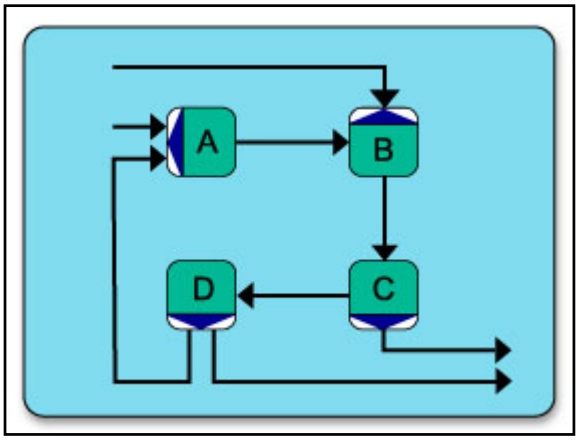
II-22. Pattern Discriminator - Implementazione A



II-23. Pattern Discriminator - Implementazione B

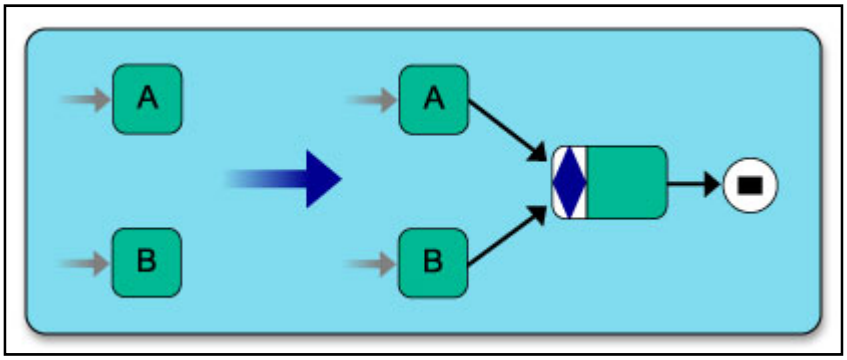
Structural Patterns

10. ***Arbitrary Cycles*** – Non ci sono limitazioni imposte dal linguaggio riguardo i cicli, di conseguenza è possibile modellare una situazione come la seguente.



II-24. Pattern Arbitrary Cycles

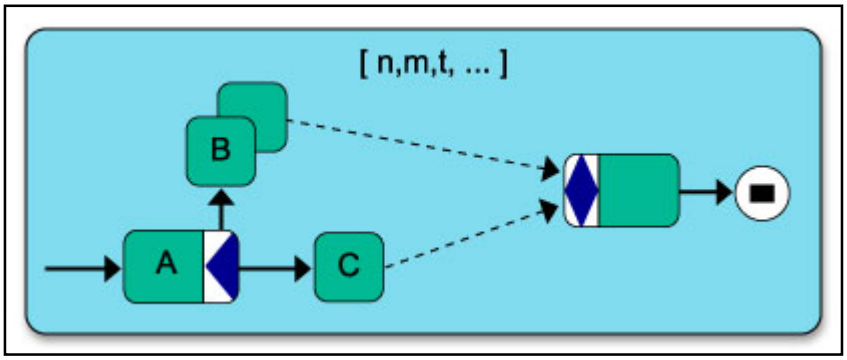
11. ***Implicit Termination*** – La terminazione implicita di un workflow è l'unico pattern che non viene realmente implementato dal linguaggio. Questo vincolo è stato scelto in modo da obbligare il modellatore a prevedere una condizione finale per ogni esecuzione del workflow. Siccome ogni ***EWF - net*** presenta una sola condizione di output, i ***Task*** che vogliamo considerare come implicitamente terminati devono essere ricollegati a questa condizione come in figura.



II-25. Pattern Implicit Termination

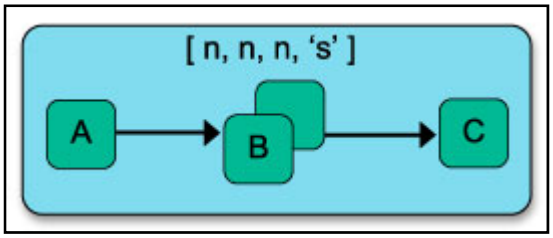
Multiple Instances Patterns

12. ***Multiple Instances Without Synchronization –*** Utilizzando un ***Task*** AND - Split e un ***Task*** OR-join è possibile implementare questo pattern, che prevede l'esecuzione di istanze multiple di un ***Task*** senza sincronizzazione con gli altri ***Task*** attivi in parallelo.



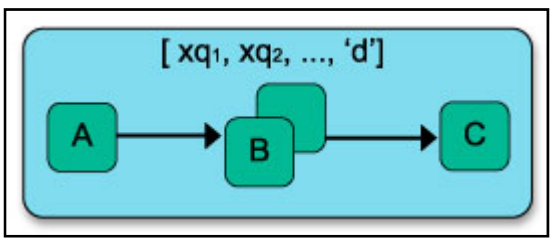
II-26. Pattern Multiple Instances Without Synchronization

13. ***Multiple Instances With a Priori Design Time Knowledge*** – Questo pattern prevede che al momento della modellazione si conosca il numero di istanze di un determinato ***Task*** per ogni sua esecuzione. Per fare questo si sfrutta un'altra proprietà dei ***Task***. E' possibile infatti indicare come ***dynamic*** o ***static*** il ***Task***, se si vuole permettere o meno che vengano create nuove istanze del task dopo la sua attivazione. In questo caso quindi il ***Task*** sarà indicato come statico.



II-27. Pattern Multiple Instances With a Priori Design Time Knowledge

14. ***Multiple Instances Without a Priori Design Time Knowledge -*** Similmente al pattern precedente si sfrutta la proprietà ***static / dynamic*** del ***Task*** in questione. Nell'esempio, al termine di A vengono valutate xq1 e xq2 (espressioni XQueries (40)) che determinano il numero (minimo e massimo) di istanze da attivare, concluse le quali viene attivato C.

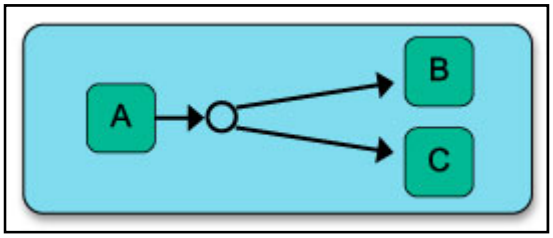


II-28. Pattern Multiple Instances Without a Priori Design Time Knowledge

15. ***Multiple Instances With a Priori Run Time Knowledge*** **–** Situazione mista tra i pattern precedenti in cui vengono utilizzate delle espressioni XQueries (40) per determinare il numero esatto di istanze da attivare. Il ***Task*** viene modellato come statico.

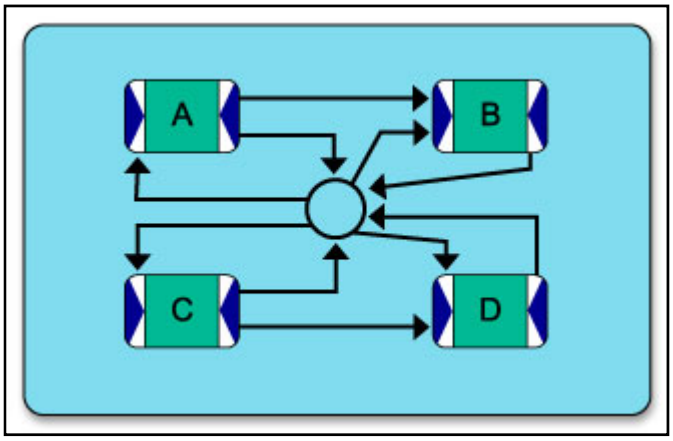
State-Based Patterns

16. ***Deferred Choice*** – Similmente all' ***Exclusive Choice***, solo uno dei ***Task*** a valle di A deve essere attivato, ma la scelta non è effettuata esplicitamente (ad esempio basandosi su dati del sistema). Viene quindi rimandata la scelta il più tardi possibile lungo il cammino, inserendo una ***condition*** che effettui lo Split.



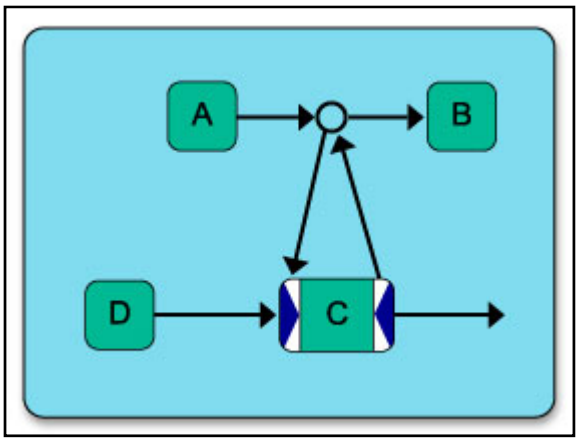
II-29. Pattern Deferred Choice

17. ***Interleaved Parallel Routing*** – Il concetto che sta alla base di questo pattern è la possibilità di modellare una situazione in cui l'ordine di esecuzione di alcune attività non sia conosciute a priori, ma nessuna di queste può avvenire in parallelo rispetto alle altre.



II-30. Pattern Interleaved Parallel Routing

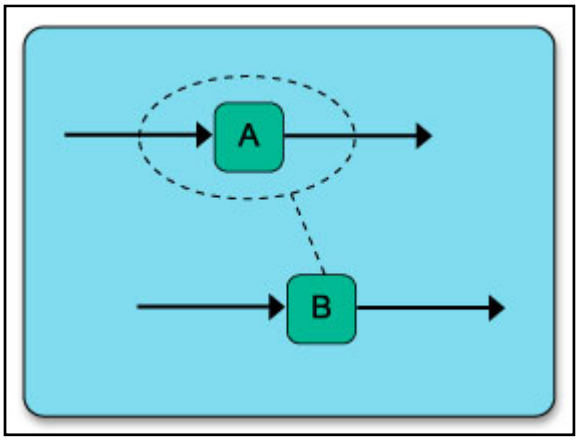
18. ***Milestone*** – In questo caso C non viene attivato se non è stato raggiunto un determinato stato "abilitante" tra A e B. Se si raggiunge il ***Task*** B allora C non può essere eseguito.



II-31. Pattern Milestone

Cancellation Patterns

19. ***Cancel Activity*** – L'esecuzione di B comporta la cancellazione del ***Task*** A. L'implementazione mostrata in figura indica come la cancellazione non prevede nessun tipo di relazione tra i due ***Task*** coinvolti.



II-32. Pattern Cancel Activity

20. ***Cancel Case*** – Stessa situazione del pattern precedente, nel caso di un generico elemento del modello come ad esempio un'istanza di workflow.

Capitolo 12. Data Perspective in YAWL

In YAWL (63) i dati, sia che si tratti di ***Data Elements*** (dati memorizzati in variabili), sia che si tratti di ***Data*** (informazioni memorizzate in documenti YAWL (63)), sono tutti definiti con un tipo (***Data Type***). Il linguaggio supporta alcuni tipi primitivi (e.g. ***boolean***, ***string***, ***long ..***) con cui è possibile definire in XML (16) dei tipi di dato complessi (elemento ***complexType***).

Data Visibility

Le variabili del sistema possono essere associate all'intera net (quindi accessibili a tutti i task del processo) oppure ad un solo task (quindi accessibili solo dal contesto di esecuzione del task stesso). La definizione di una variabile comprende anche la dichiarazione del tipo di utilizzo previsto tra le seguenti opzioni:

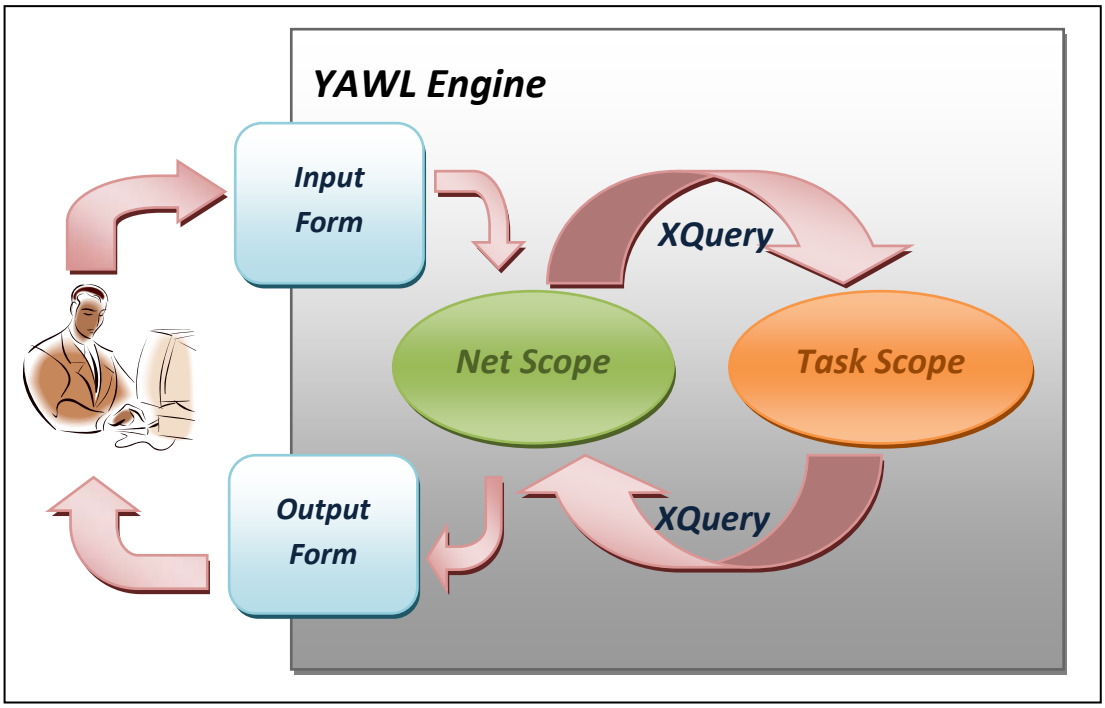
- ***Input only –*** la variabile riceve i dati da una variabile esterna allo scope;
- ***Output only –*** la variabile fornisce i dati ad una variabile esterna allo scope;
- ***Input & output –*** la variabile riceve e fornisce i dati all'esterno dello scope;
- ***Local variable –*** la variabile viene utilizzata esclusivamente per manipolazioni interne allo scope.

Data Transfer

Il passaggio dei dati all'interno dell'ambiente di esecuzione avviene esclusivamente tra un task e la net o viceversa (non esiste quindi un passaggio diretto di informazioni tra due task). La definizione del passaggio dei dati si avvale di XQuery (40) che accoppiano gli ***input parameters*** e ***output parameters*** del task alle variabili della net, lasciando al sistema la gestione del trasferimento di informazioni.

Lo scambio di dati con l'ambiente esterno viene gestito da YAWL (69)automaticamente attraverso la generazione automatica di apposite form per richiedere l'inserimento dei dati all'utente. Tale scambio di dati è previsto solo per le variabili della net che non siano ***local variable***.

Il trasferimento dei dati (sia interno che esterno) avviene quindi in modo automatico e si serve di alcune tabelle di decomposizione che contengono gli accoppiamenti tra le variabili (di net e di task) sotto forma di XQuery (40).



II-33. Data Transfer in YAWL

Il sistema fornisce alcuni servizi per facilitare la modellazione della scambio dei dati, tra cui la segnalazione automatica della mancanza di variabili utili al trasferimento stesso e la generazione di XQuery (40) corrette e compatibili con il YAWL Engine (69).

Data Related

Come già visto nel Capitolo 11, l'attivazione di un ramo in uscita dai task OR - Split e XOR - Split dipende dal verificarsi di una condizione ad esso associata. Questa condizione (***branching condition***) si definiscono come condizioni booleane XPath (39) nei dettagli di flusso del task cui ci si riferisce. La ***branching condition*** sarà quindi parametrica secondo delle variabili (sono permesse solo variabili della net). La valutazione dell'espressione XPath (39) avviene a carico dell'Engine, che attiverà i rami la cui ***branching condition*** associata risulta vera (o il ramo di default se tutte sono false).

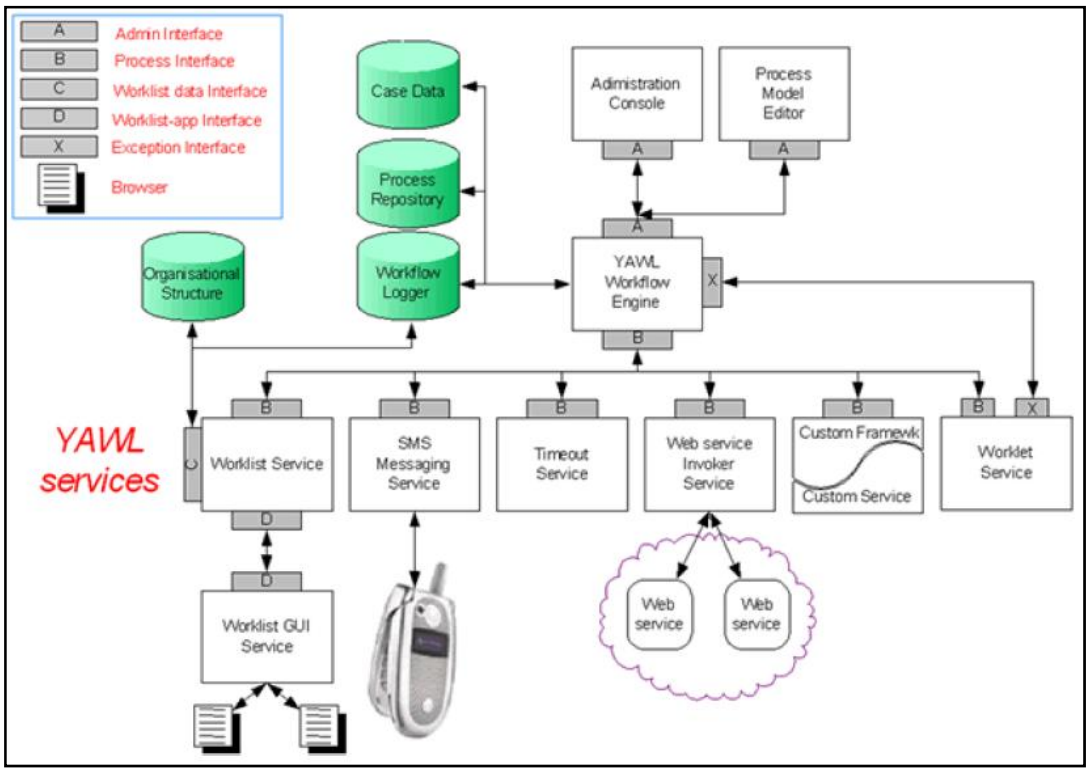
Capitolo 13. YAWL Engine

Interfaccia

L'interfaccia dell'Engine, cui si accede previa autenticazione, è suddivisa in quattro sezioni:

- ***Administrate*** – Da questa sezione è possibile
  - Gestione delle specifiche di workflow (caricamento, disattivazione..)
  - Gestione risorse quali utenti del sistema e altre macchine interfacciate ad esso
  - Gestione dei ruoli, categorizzazione dei livelli di accesso e di allocazione del lavoro in base ai ruoli o all'identificazione del nome della risorsa
  - Consultazione di report statistici sul sistema
- ***Workflow Specifications* –** sezione da cui è possibile lanciare / terminare un 'istanza di processo e avere informazioni su quelle già attive
- ***Available Work* –** task disponibili pronti per essere attivati dall'utente
- ***Checked Out Work* –** task attivati e in attesa di inserimento dati da parte dell'utente
- ***Logout* –** uscita dal sistema

Architettura e Servizi



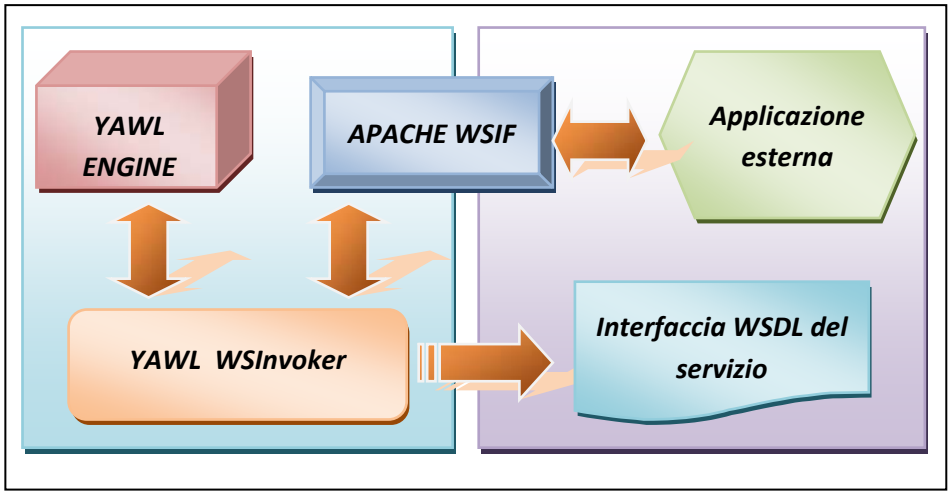
II-34. Architettura di YAWL

Com'è possibile vedere dallo schema presentato in figura II-34. Architettura di YAWL (69), il sistema risulta essere piuttosto modulare. Attraverso l'analisi delle interfacce previste tra i moduli, ne verrà analizzata a grandi linee la struttura:

- ***Admin Interface*** – L'interfaccia di amministrazione permette di interagire con l'Engine vero e proprio in modo da effettuare le operazioni che abbiamo visto essere disponibili sull'interfaccia utente per l'amministratore. Possiamo notare che il ***YAWL workflow Engine*** presenta un forte accoppiamento con i database, in quanto si occupa di gestire tutte le informazioni del sistema.
- ***Process Interface –*** Questa interfaccia permette allo ***YAWL workflow Engine*** di invocare i vari servizi interni del sistema.
- ***Worklist data interface –*** Interfaccia con cui il ***worklist service*** dialoga con i database per poter eseguire i task delle specifiche di workflow
- ***Worklist – app interface –*** Interfaccia con cui la ***worklist GUI service*** scambia informazioni con la ***worklist service*** in modo da permettere il trasferimento dati da / per l'esterno
- ***Exception interface –*** Interfaccia utilizzata per la gestione delle eccezioni attraverso il ***worklet Service***

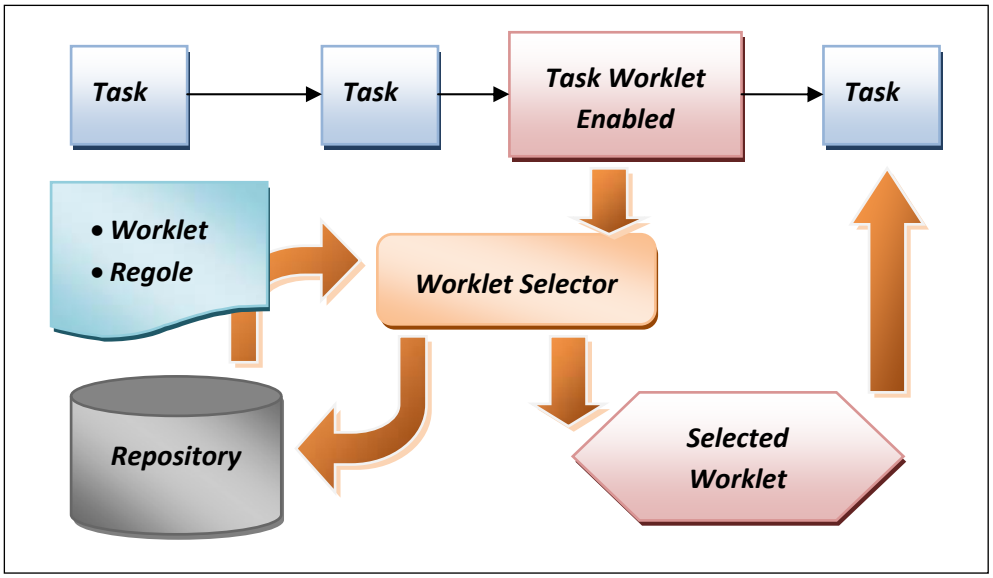
Si nota quindi che YAWL (69) ha un approccio orientato fortemente ai servizi. In particolare nell'architettura notiamo i seguenti servizi integrati:

- ***Worklist Service*** – di esecuzione dei task
- ***SMS Messaging Service*** – invio/ricezione SMS
- ***Timeout Service*** – Web service in grado di gestire le deadline e quindi i timeout in fase di allocazione /de allocazione delle attività.
- ***Web Service Invoker Service*** – permette al sistema di interagire con Web Service (8) esterni per eseguire il lavoro corrispondente di un task. In particolare questo viene "decomposto" in un invocazione al servizio esterno utilizzando l'interfaccia WSDL (67) presente nella descrizione del processo. In particolare l'esecuzione del task comporta:
  i. All'attivazione dell'istanza del task, lo ***YAWL WSInvoker*** recupera l'interfaccia WSDL (67) del task e utilizzando il ***WSIF*** (***Apache Web Services Invocation Framework***) (70)ricava le informazioni necessarie all'invocazione del servizio
  ii. Il task passa da ***enabled*** a ***processing*** e avviene il passaggio all'applicazione esterna
  iii. Il servizio esterno termina e ripassa il controllo al task che passa in modalità ***completed***.



II-35. YAWL WSInvoker

- ***Custom Framework / Custom Service*** – Permette l'interfacciamento con servizi creati dall'utente che dialoghino con il ***YAWL Engine*** con messaggi XML (16) via HTTP (44). Non vi sono ovviamente limitazioni sul linguaggio con cui viene implementato il servizio, e l'unico vincolo è quello di definire una URL (71) su cui questo deve rispondere.
- ***Worklet Service*** – Una ***worklet*** si definisce come "*processo a se stante, definito in YAWL, che agisce come sub net per una workitem*" (72). In pratica quando non è possibile definire a priori (o si vuole permettere maggiore flessibilità) ogni possibile situazione nella specifica di workflow, si definisce un task ***worklet enabled***. In questo modo il servizio si preoccupa di selezionare la ***worklet*** che deve essere eseguita al posto di quel task, attraverso delle regole definite nella ***worklet*** stessa. Per questo scopo il servizio si compone in
  - ***i.*** ***Worklet Selector Service –*** Servizio che si occupa di recuperare le ***worklet*** associate ad un task ***worklet enabled***, selezionare quella più opportuna al caso in base ad un insieme di regole (espresse in XML (16)) e sostituire temporaneamente il task con essa.



II-36. Worklet Selector

ii. ***Worklet Exception Service*** – In YAWL (63) le eccezioni sono gestite al pari delle altre attività, con la differenza che non sono state pensate in fase di progettazione del workflow. Questo servizio sfrutta quindi la flessibilità delle ***worklet*** (72) per gestire il verificarsi dell'eccezione, caricando il worklet opportuno (se non ne trova l'eccezione viene ignorata). Attraverso le regole è possibile definire diversi tipi di eccezioni a seconda che la causa sia :

- Un vincolo non atteso (prima e/o dopo l'esecuzione del task)
- Un evento esterno
- Timeout
- L'annullamento di un task
- Risorsa non disponibile
- Violazione di un vincolo (durante l'esecuzione del task).

Capitolo 14. newYAWL

Negli anni successivi alla concezione di YAWL (63) l'utilizzo del sistema ha portato alla luce alcune carenze del linguaggio, spingendo i ricercatori a potenziarlo e revisionarlo. E' così che nel 2007 la YAWL foundation (22) pubblica l'articolo di presentazione / introduzione di newYAWL (73).

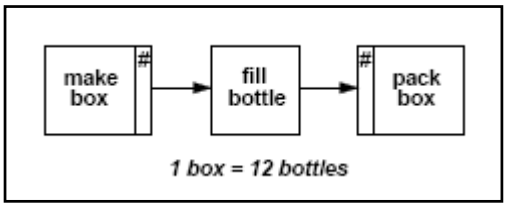
Gli obiettivi che si sono voluti raggiungere sono stati

- L'introduzione di un supporto per i pattern non ancora implementati dal sistema ma risultati importanti per la sua espressività nelle varie prospettive.
- Fornire maggiore supporto alle prospettive dei dati e delle risorse oltre che ai control flow.
- Definire in termini di Coloured Petri Nets (64) un modello eseguibile relativo alla semantica di ogni costrutto del sistema.
- Definire un modello astratto sintattico di ogni elemento del sistema.

Scopo di questo capitolo sarà presentare brevemente le novità introdotte con newYAWL (73).

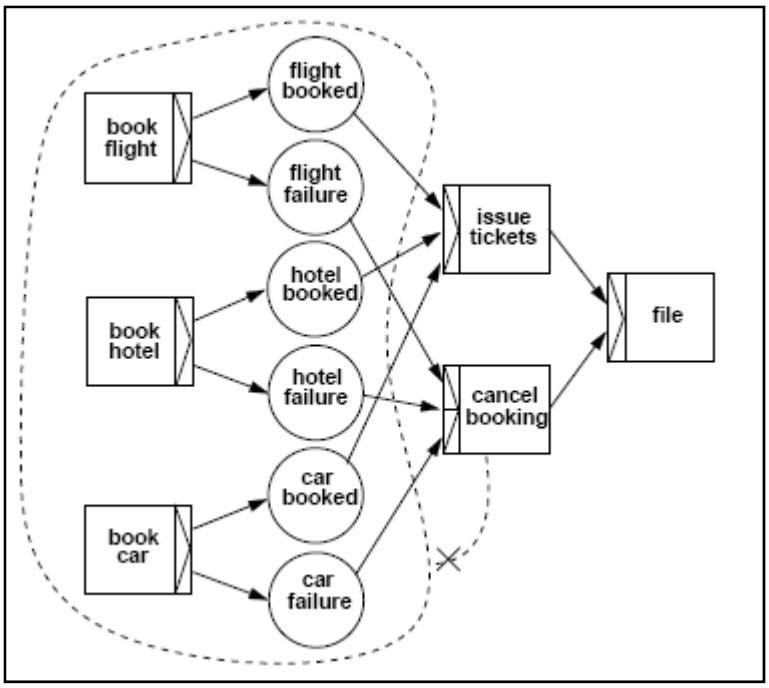
Control –flow perspective

1. ***Thread Split / Thread Merge*** – Questi due ***Task*** permettono di modellare lo Split /merge del ***thread*** di controllo in multipli ***thread*** concorrenti, precisandone il numero a design time. Nell'esempio in figura II-37 ogni ***thread*** che attraversa il ***task*** *make box*, viene diviso in 12 ***thread*** (uno per ogni bottiglia del collo) per poi venire riunificato in un unico ***thread*** per l'attivazione di *pack box.*



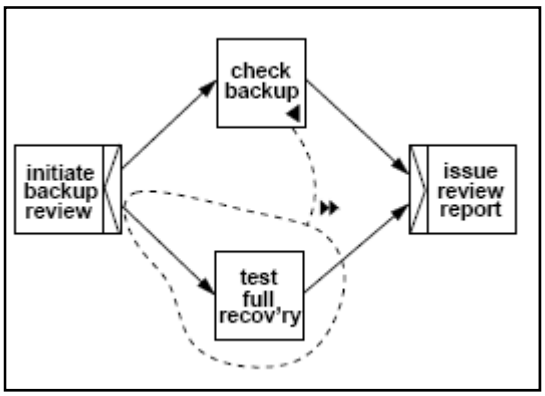
II-37. Pattern Thread Split / Thread Merge

2. ***Partial Join*** – Costrutto che permette di attivare il ***Task*** quando m degli n percorsi in ingresso sono attivi (M out of N Join). Nell'esempio *cancel booking* attiva la cancellazione delle attività di prenotazione quando almeno uno dei suoi ingressi (fallimento di una delle operazioni) si attiva.



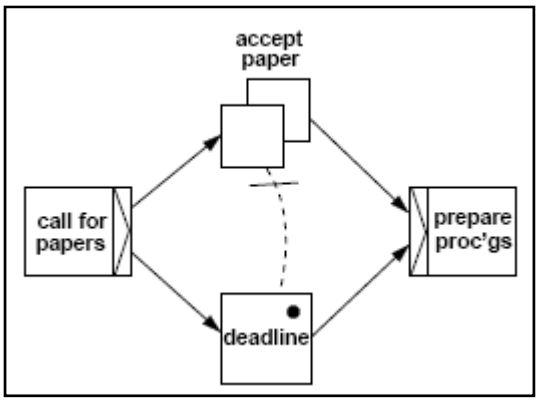
II-38. Pattern Partial Join

3. ***Structured Loop*** – Consente la ripetizione di un ***Task*** in funzione di un test condizionale. Nell'esempio in figura II-39 *check backup* viene eseguito per ogni backup esistente.



II-39. Pattern Structured Loop & Completion Region

4. ***Completion Region*** – Costrutto che supporta il completamento forzato degli elementi racchiusi dalla regione stessa. Nell'esempio in figura II-39 *test full recovery* viene forzato al completamento non appena (tutte le iterazioni di) *check backup* sono completate, in modo da attivare immediatamente *issue review report*.
5. ***Persistent Trigger*** / ***Transient Trigger*** – Questi costrutti permettono di attivare un ***Task*** in seguito al verificarsi di un evento. La natura dell'attivazione può essere rispettivamente persistente o transitorio. Nell'esempio in figura II-40 *deadline* viene attivato in maniera persistente.



II-40. Pattern Persistent Trigger /Transient Trigger & Disablement Arc

6. ***Disablement arc –*** Questo elemento impedisce alla sua attivazione che un ***Multiple Instance Task*** generi ulteriori istanze. In figura II-40 *deadline* blocca quindi le successive istanze di *accept paper.*

Data perspective

La prospettiva dei dati è stata maggiormente definita in modo da non dover ricorrere a supporti informatici di terze parti per la determinazione della gestione di problematiche quali concorrenza o manipolazione complessa dei dati.

1. ***Scopes*** – Possibilità di legare gli elementi della prospettiva dei dati a diversi scope quali :
    a. ***Global*** - disponibilità per tutti gli elementi del processo;
    b. ***Folder*** - disponibilità per gli elementi del processo legati ad una precisa locazione;
    c. ***Case -*** relativo ad una determinata istanza del processo;
    d. ***Block –*** relativo ad un sottoprocesso;
    e. ***Scope –*** disponibilità solo per un determinato sottoinsieme di elementi di un processo;
    f. ***Task –*** relativo ad un task;
    g. ***Multiple Instance –*** relativo ad un'istanza di un task a istanze multiple.
2. ***Formal Parameters –*** parametri formali per il trasferimento di informazioni tra i vari costrutti basato su un approccio a funzioni.
3. ***Link Conditions –*** condizioni in base ai quali i rami uscenti da XOR - Split Task e OR - Split Task si debbano o meno attivare.
4. ***Pre / PostConditions –*** condizioni in ingresso e in uscita per Task e processi.
5. ***Locks –*** costrutto che permette di garantire accesso esclusivo a determinate risorse da parte di un Task.

Resource Perspective

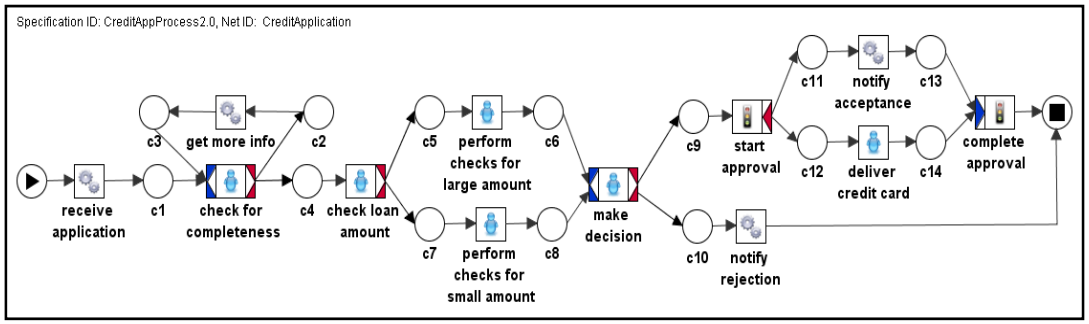
La prospettiva delle risorse permette di ottimizzare il controllo sia sulla distribuzione del lavoro agli utenti, sia su come le risorse stesse debbano trasferirsi lungo il processo. Per ogni task è possibile descrivere

1. ***Interaction Strategy*** in cui definire come il lavoro debba essere comunicato all'utenza, come deve essere stabilita la sua accettazione e come dovrà essere definito il tempo necessario alla lavorazione.
2. ***Routing Strategy*** per definire gli utenti potenziali cui poter sottoporre il lavoro. E' possibile definire a questo scopo regole in base a :
    a. nome dell'utente;
    b. ruolo dell'utente;
    c. criteri per rinviare la decisione a run - time
    d. vincoli relativi alle capacità che devono avere gli utenti;
    e. struttura organizzativa in cui l'utente deve risiedere;
    f. risultati di precedenti esecuzioni del processo;
    g. Vincoli predefiniti quali
        i. ***Retain familiar*** **–** riferirsi all'utente che ha effettuato l'ultima operazione
        ii. ***Four eyes principle*** **–** riferirsi un'utente diverso da colui che ha effettuato l'ultima operazione
        iii. ***Random allocation*** **–** allocazione casuale tra quelli potenziali
        iv. ***Round robin al location*** **–** bilanciamento tendenzialmente equo tra gli utenti
        v. ***Shortest queue al location*** **–** allocazione all'utente con la coda di lavoro più corta tra quelli potenziali.

Capitolo 15. Esempio di Processo in YAWL

Questo capitolo ha lo scopo di presentare un caso pratico di progettazione di un processo in YAWL (63), illustrando la modellazione delle control - flow, data e resources perspectives prima e l'esecuzione del processo nell'ambiente di supporto di YAWL.

L'esempio riportato in figura II-41si basa sulla demo disponibile sul sito della YAWL foundation (22) relativo alla richiesta di una carta di credito.



II-41. Control Flow Credit Application

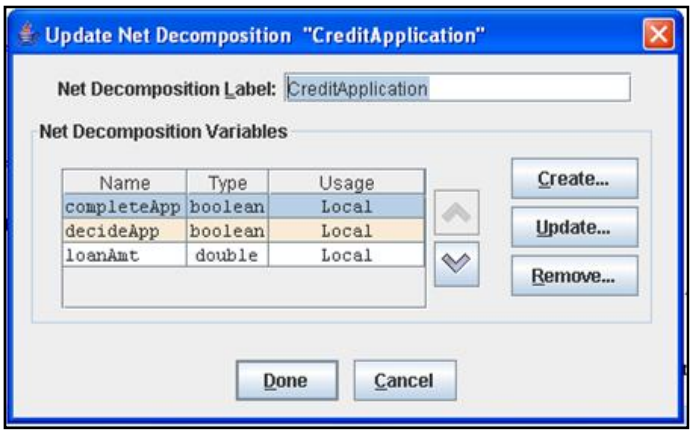
Control – flow

Il processo inizia quando viene ricevuta una richiesta di emissione da parte di un utente (attività automatica). Ricevuta la richiesta, un impiegato ne controlla la completezza (attività umana) ed eventualmente viene emanata una richiesta per ricevere le informazioni mancanti (XOR - Join / XOR - Split) . Successivamente vengono effettuati diversi controlli sul conto del richiedente. Si noti che dopo il task *check loan account*, a seconda del valore del conto si attiva un ramo diverso del flusso di controllo. Stessa cosa avviene nel caso di *make decision*, a seconda che la richiesta venga accettata o respinta. Possiamo notare che *start approval* e *complete approval* sono due task di tipo ***routing*** e sono modellati rispettivamente come AND - Split e AND - Join.

Data Perspective

Creiamo tre variabili di net:

- ***loanAmt*** – (double) è l'ammontare del prestito
- ***decideApp*** **–** (boolean) per indicare se la richiesta è approvata
- ***completeApp*** **–** (boolean) per indicare se la richiesta è completa

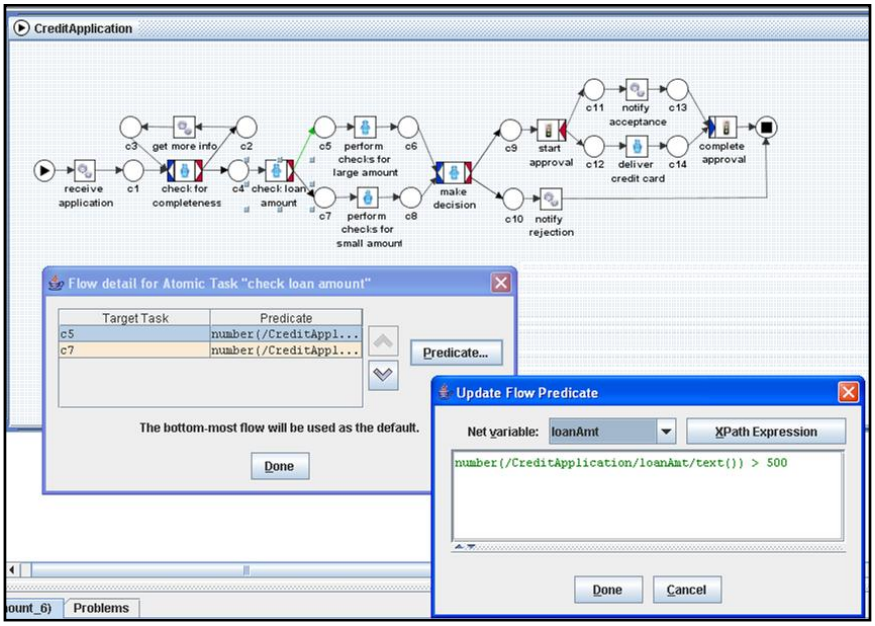


II-42. Screenshot Net Variables

Queste variabili vengono usate nei task XOR - Split per decidere quali rami in uscita vengono attivati. Ad esempio in *check loan amount* l'espressione da testare per proseguire alla condizione *c5* sarà

number(/CreditApplication/loanAmt/text()) >500

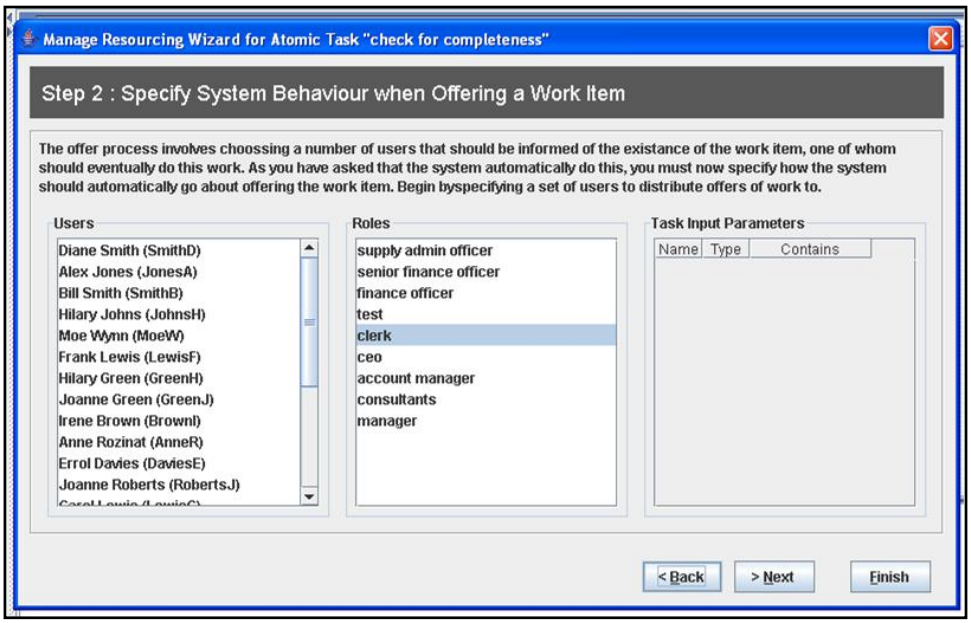
Notiamo quindi che si tratta di un'espressione booleana dove individuiamo in XPath la variabile di net ***loanAmt*** e verifichiamo che il suo valore sia maggiore di 500.



II-43. Screenshot Flow details *check loan amount*

Resource Assignments

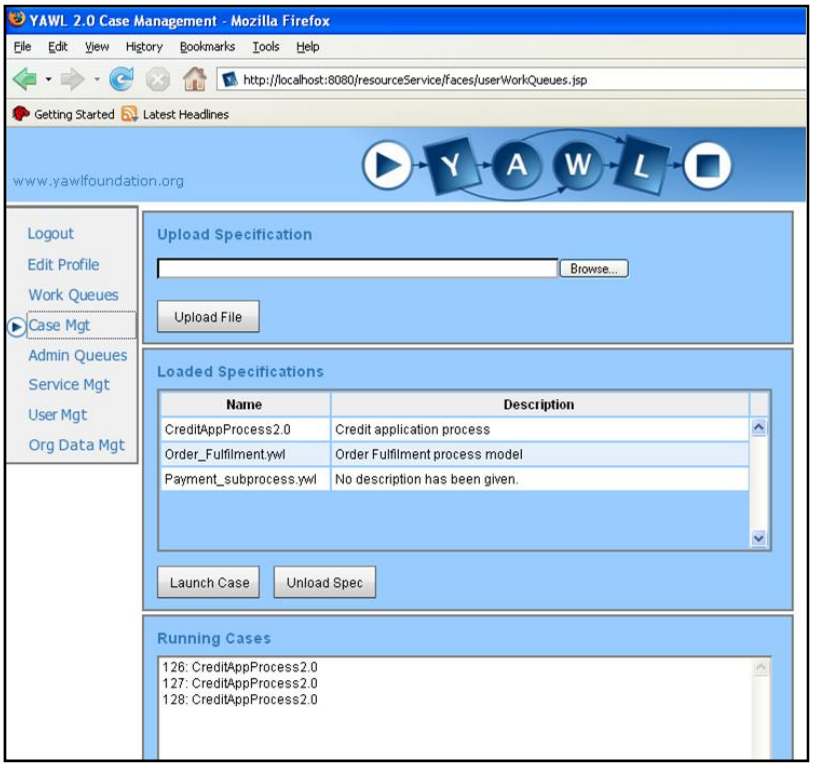
Utilizziamo ora l'assegnamento delle risorse sui vincoli in base al ruolo dell'utente. In particolare abbiamo individuato diversi task che richiedono decisioni umane, e vogliamo permetterne l'esecuzione ai ruoli ***Clerk*** e ***Manager*** (quest'ultimo solo per il task *make decision*).



II-44. Screenshot wizard Resource Perspective

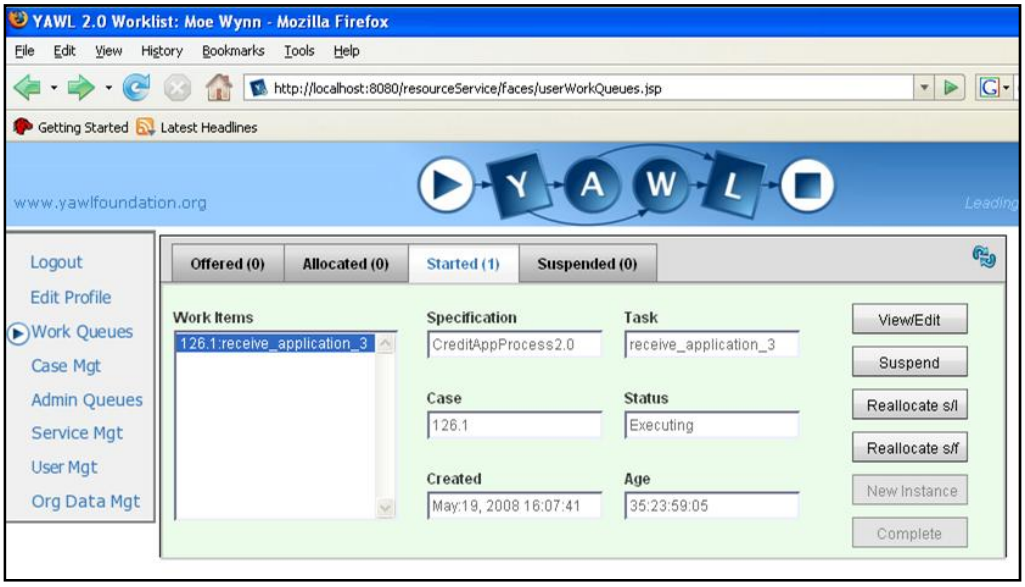
Esecuzione del processo

Una volta modellate le prospettive del control - flow, dei dati e delle risorse, il modello può venire caricato dal front - end di amministrazione. Caricato il modello, può venire lanciato un case (ovvero un'istanza del processo).



II-45. Screenshot Case Management

Per ogni case avviato compariranno (tra i workitem disponibili degli utenti con ruolo *Clerk* e *Manager*) le istanze di task che necessitano dell'intervento umano. Come già visto nel Capitolo 13 è possibile gestire l'allocazione del lavoro ridistribuendolo oppure sospendendolo a seconda dell'esigenza.



II-46. Screenshot Worklist

Sezione III. Confronto tra i due approcci

Capitolo 16. Obiettivo

Una volta introdotti i due approcci, possiamo passare allo scopo vero e proprio di questa tesi, ovvero confrontarli in modo da poter capire quali sono i punti di forza e di debolezza di entrambi. Tale confronto può essere inteso come una guida per la scelta di uno o dell'altro approccio a seconda delle esigenze aziendali, ma al tempo stesso può essere uno spunto per la loro evoluzione in modo da poter sviluppare una soluzione più completa e potente al servizio del Management.

In primo luogo verrà presentato un caso di studio reale che verrà mappato e simulato con entrambi gli approcci già presentati in questa tesi. Secondo passo sarà quello di evidenziare aspetti negativi e positivi di entrambi gli approcci per quanto riguarda il design e le potenzialità offerte nell'esecuzione / simulazione dei processi.

Capitolo 17. Il caso di studio

Come già anticipato il confronto avverrà su un caso di studio reale, ovvero un'azienda pilota del consorzio "100% italiano", la S.A.P.A.F s.n.c. di Andrea Calistri. L'azienda, già oggetto di studio della tesi "Modellazione BPMN per piattaforme di Business Process management: esperienze nel settore della pelletteria" (74) del mio collega Davide Stefani e della pubblicazione " (75) curata da Francesco Marcelloni e Mario Giovanni C. A. Cimino all'interno della collana "Ricerca Trasferimento Innovazione" (76)della Regione Toscana, è simbolo di una realtà in cui arte, mestiere e tecnologia si fondono in ottica del perseguimento della qualità. Errore comune nella realtà imprenditoriale di oggi è infatti credere che l'ICT non possa fornire un vero e proprio valore aggiunto alle PMI (77) da cui l'industria italiana è prevalentemente formata, e la S.A.P.A.F è un valido esempio di come proprio le aziende rete siano le maggiori beneficiarie dello sfruttamento delle ICT per integrazione dei sistemi in ottica di una qualità sempre crescente.

Riportiamo di seguito la descrizione dell'azienda in questione come presentato nel capitolo 6 della pubblicazione "INNO.PRO.MODA" (75):

L'azienda è caratterizzata dalla presenza di diverse aree operative: l'Area Ricerca Sviluppo e Marketing, l'Ufficio Stile, l'Ufficio Tecnico, il Reparto Elaborazione, il Reparto Produzione, il Reparto Assemblaggio ed il Magazzino Prodotto Finito. L'Area Ricerca Sviluppo e Marketing è dedicata a cogliere gusti e tendenze, fare ricerche di mercato, a gestire il contatto con i clienti e a monitorare le fiere. In quest'area vengono anche sviluppate soluzioni per aspetti particolari dei prototipi, es. composizioni grafiche di colore.

La nascita del prodotto avviene nell'Ufficio Stile, dove si realizzano le richieste del cliente in formato cartaceo, creando lo schizzo di ciò che si vuole realizzare, con indicazione delle forme, materiali, dettagli, colori e caratteristiche d'importanza per la linea produttiva. Tutte le informazioni che provengono dall'Ufficio Stile, sono messe assieme, per preparare lo sviluppo, che avviene nell'Ufficio Tecnico. In seguito all'ordine di produzione e prima realizzazione dello schizzo, viene creta una scheda verde che contiene l'idea originaria, detta anche specifica, stabilita con il cliente, la bozza del prodotto ed eventuali note aggiuntive.

Dopodiché, per ogni modello, viene creato un fascicolo tecnico, la scheda gialla, che contiene il capitolato del progetto, ossia tutte le informazioni relative al modello, dal disegno tecnico (che, rispetto alla bozza, contiene dettagli tecnici come, per esempio, il tipo di filo da utilizzare per l'interno e l'esterno, il colore e lo spessore di esso, la lunghezza del cucito, il tipo di ago da utilizzare, gli accessori – come devono essere, il tipo e dove devono essere applicati, ecc.), a tutte le informazioni utili per la realizzazione del prodotto. La scheda gialla accompagna il prodotto per tutta la sua realizzazione e serve da guida per gli operatori addetti alla produzione dello stesso.

Tutte queste informazioni sono poi inviate al Reparto Elaborazione in cui è presente un'area di "Progettazione CAD" che ha il compito di creare una rappresentazione digitale del prodotto da realizzare. Questa rappresentazione consente innanzitutto di ideare le diverse

sagome di cui si compone il prodotto. Tali sagome possono essere generate tramite un normale plotter su carta, oppure tramite un plotter particolare direttamente su uno specifico tipo di materiale.

Generata la rappresentazione digitale, ha inizio la fase di creazione del primo prototipo (o modello) che generalmente viene realizzato utilizzando una salpa o canapino, materiali creati dagli scarti e di consistenza simile a quella dei materiali finali. Ciò consente di avere un prodotto molto simile a quello originale, su cui annotare eventuali correzioni che vengono proposte durante una riunione tra l'ufficio stile ed il committente (o stilista) effettuata proprio per stabilire eventuali modifiche all'idea originaria.

Nel caso in cui siano necessarie delle modifiche, l'ufficio stile provvede a modificare la scheda verde e la scheda gialla ed avviare una nuova creazione digitale. Nel caso in cui il prototipo soddisfi i requisiti del cliente, si procede con la creazione di una distinta base, realizzabile attraverso le informazioni ricevute dalla progettazione CAD, dettagliata di ogni singolo pezzo con relativa forma e dimensione. Dalla distinta base un altro software è in grado di calcolare il costo, individuando la quantità necessaria di ogni singola parte e definendo il tempo delle varie fasi di lavorazione (taglio, preparazione componenti, assemblaggio, imballaggio e spedizione). Basandosi sul costo, viene formulato un prezzo che viene proposto al cliente. L'accettazione da parte del cliente dà il via alla fase di creazione del prototipo reale (o campione), ovvero un prototipo realizzato in materiali stabiliti che riporta le correzioni definitive.

Al termine di questa fase si interagisce nuovamente con il committente (stilista) che stabilisce se il prodotto risponde globalmente al gusto della tipologia del cliente al quale è destinato, concordando eventuali soluzioni tecniche specifiche.

In caso di approvazione, inizia la Fase di Produzione, che viene dettagliata in una scheda rossa, o "scheda ordinativo produzione". Come già scritto, durante le varie fasi di lavorazione è sempre la scheda gialla il riferimento tecnico, mentre la scheda rossa contiene solo l'ordinativo di produzione.

I dati del CAD arrivano direttamente dall'implementazione effettuata nel Reparto Elaborazione ad un plotter orizzontale, che effettua la fase di Taglio. In particolare, l'operatore che riceve il foglio di lavorazione, sceglie da un database di forme il modello da tagliare. Il plotter orizzontale propone il materiale da tagliare e il tipo di oggetto da tagliare. Dopo aver steso le pelli sul piano di taglio, è possibile tramite il plotter, proiettare le forme sul tessuto steso sul piano, consentendo di non utilizzare le "fustelle" (oggetti in metallo che consentono di tagliare le forme sui materiali).

Tramite una fotocamera digitale collegata al plotter, è inoltre possibile rilevare le informazioni sulla pelle, le sue dimensioni ed il materiale di cui è composta. Nel caso in cui le pelli non siano buone, l'operatore valuta se siano o meno trattabili od utilizzabili per altre lavorazioni e depositale pelli nella zona adibita alle pelli da scartare o in quella delle pelli da

trattare, stendendo così una nuova pelle sul piano del plotter e rieseguendo la valutazione tramite fotocamera digitale.

In caso di assenza di problemi, le parti vengono aggiustate sul piano, ed il plotter riesce a tagliare la pelle ottimizzando la riduzione degli scarti. Il taglio è effettuato da una lama oscillante che vibra durante l'avanzamento. E' possibile anche effettuare dei fori o dei segni per indicare dove applicare eventuali accessori.

Eseguito il taglio, i pezzi vengono numerati per garantire una combinazione delle parti con colore uniforme; questo perché, ad esempio, i colori di molti materiali naturali, come la pelle di pitone, possono variare di metro in metro. Ha inizio, a questo punto, la fase di Preparazione Componenti.

Gli operatori che ricevono i componenti, rispettando le direttive della scheda gialla stabiliscono le lavorazioni che sono necessarie per rendere il componente pronto per l'assemblaggio.

Scarnitura ed affinatura

Il pezzo può avere lo stesso spessore totale o spessori diversi tra corpo e bordo. Queste fasi permettono di attribuire ai vari componenti lo spessore richiesto.

Scarnitura: consiste nell'assottigliamento ai bordi dei pannelli in pelle
Affinatura (o spaccatura) : consiste nell'assottigliamento dello spessore dell'intero pannello di pelle.

Incollaggio

In questa fase vengono incollati o pezzi di pelle tra loro o accessori su pezzi di pelle utilizzando collanti naturali che vengono spruzzati sula pelle e lasciati asciugare. Nella zona adibita all'incollaggio, sono presenti aspiratori per evitare il deposito di polvere.

Coloritura

In questa fase vengono effettuate delle colorazioni sugli accessori o su parti della pelle (per esempio ai bordi). Questa fase viene eseguita a mano. A seconda delle richieste registrate nella scheda gialla, possono essere effettuate anche delle tamponature, per dare un effetto di invecchiamento al prodotto.

Al termine di queste fasi, viene effettuato il controllo del componente. Nel caso in cui non siano rispettate le specifiche, verrà individuata, se possibile, una rilavorazione da effettuare, altrimenti, il componente viene scartato. Se il componente passa il controllo viene avviato alla fase di pre - assemblaggio ovvero quella fase, precedente all'assemblaggio, dove si accoppiano i componenti contraddistinti dallo stesso numero.

Eseguita la fase di pre - Assemblaggio, si determina se occorre o meno passare alla fase di cucitura eseguita prevalentemente dalle cucitrici.

Terminata questa fase, osservando la scheda gialla, viene deciso se l'assemblaggio debba o meno essere assegnato ad un'azienda esterna. In caso affermativo, tutti i componenti vengono spediti all'azienda incaricata, altrimenti vengono inviati al Reparto Assemblaggio, dove gli operatori addetti li assemblano, come indicato dalla scheda gialla, realizzando il prodotto finito da depositare in magazzino.

Nel magazzino Prodotto Finito viene eseguito il controllo finale (controllo di qualità e quantità), l'imballaggio e la spedizione.

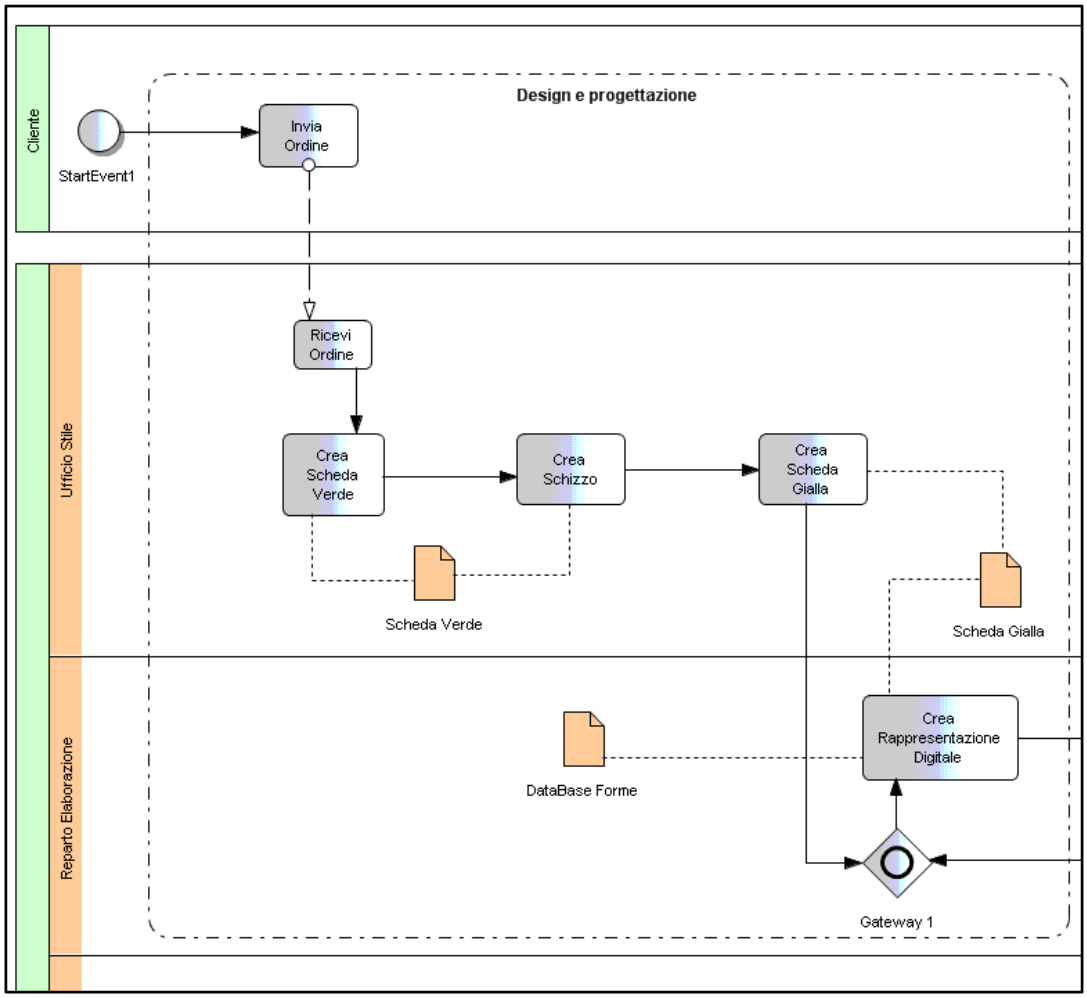
Il controllo del prodotto finito, oltre a far riferimento al manuale della qualità, verifica il rispetto delle specifiche descritte sulla scheda rossa e sulla scheda gialla. Nel caso di non conformità, viene identificato il problema, viene effettuata un'analisi per stabilire il trattamento da effettuare, viene effettuato il trattamento deciso e viene eseguita la rilavorazione. Se il ciclo ha esito positivo, il prodotto è pronto per l'imballaggio e la spedizione, altrimenti viene valutato se è possibile applicare un'ulteriore rilavorazione.

In caso di esito positivo dei controlli, viene eseguita la fase di imballaggio in cui il prodotto viene confezionato ed è pronto per la spedizione.

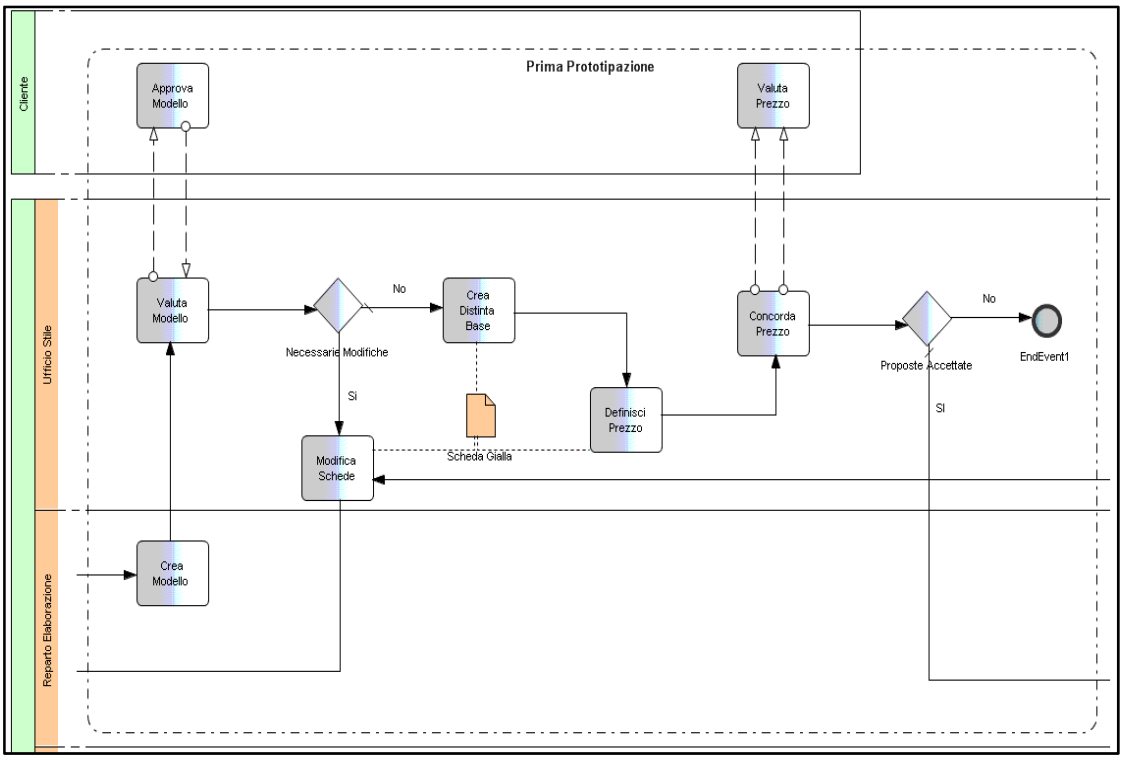
Capitolo 18. Mappatura dei processi attraverso BPMN

La descrizione del ciclo produttivo di una borsa riportata nel capitolo precedente è stata utilizzata per mappare i processi attraverso l'approccio BPMN (1). Riportiamo di seguito i diagrammi di tale mappatura.

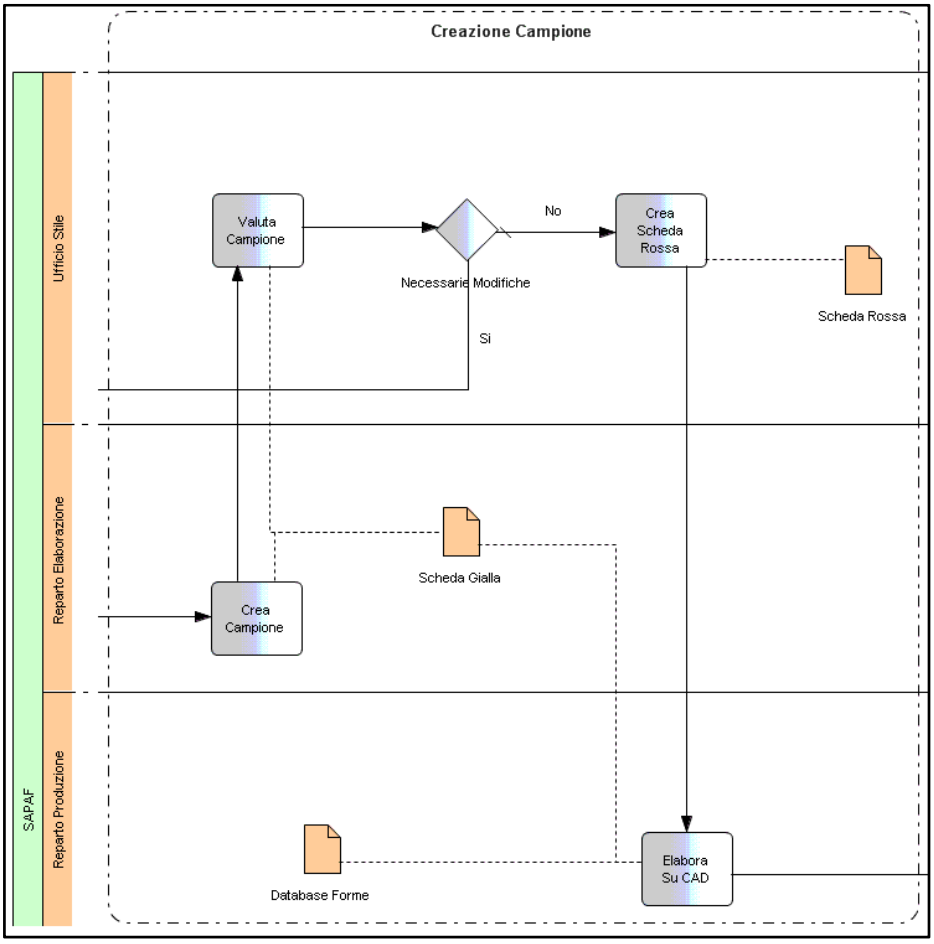
Il primo diagramma è quello top level. Essendo di dimensioni notevoli abbiamo preferito suddividerlo in 6 parti.



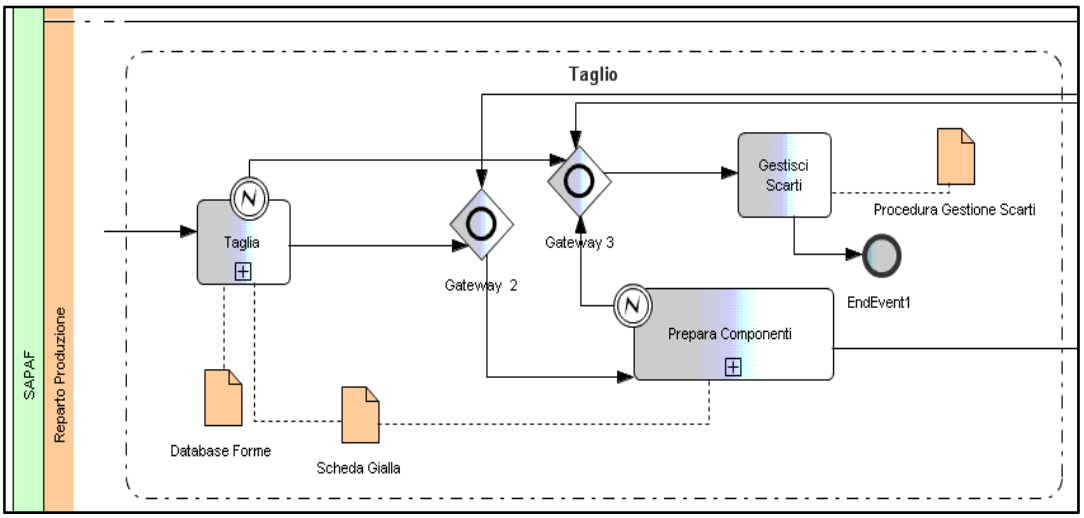
III-1. BPMN Top Level 1/6



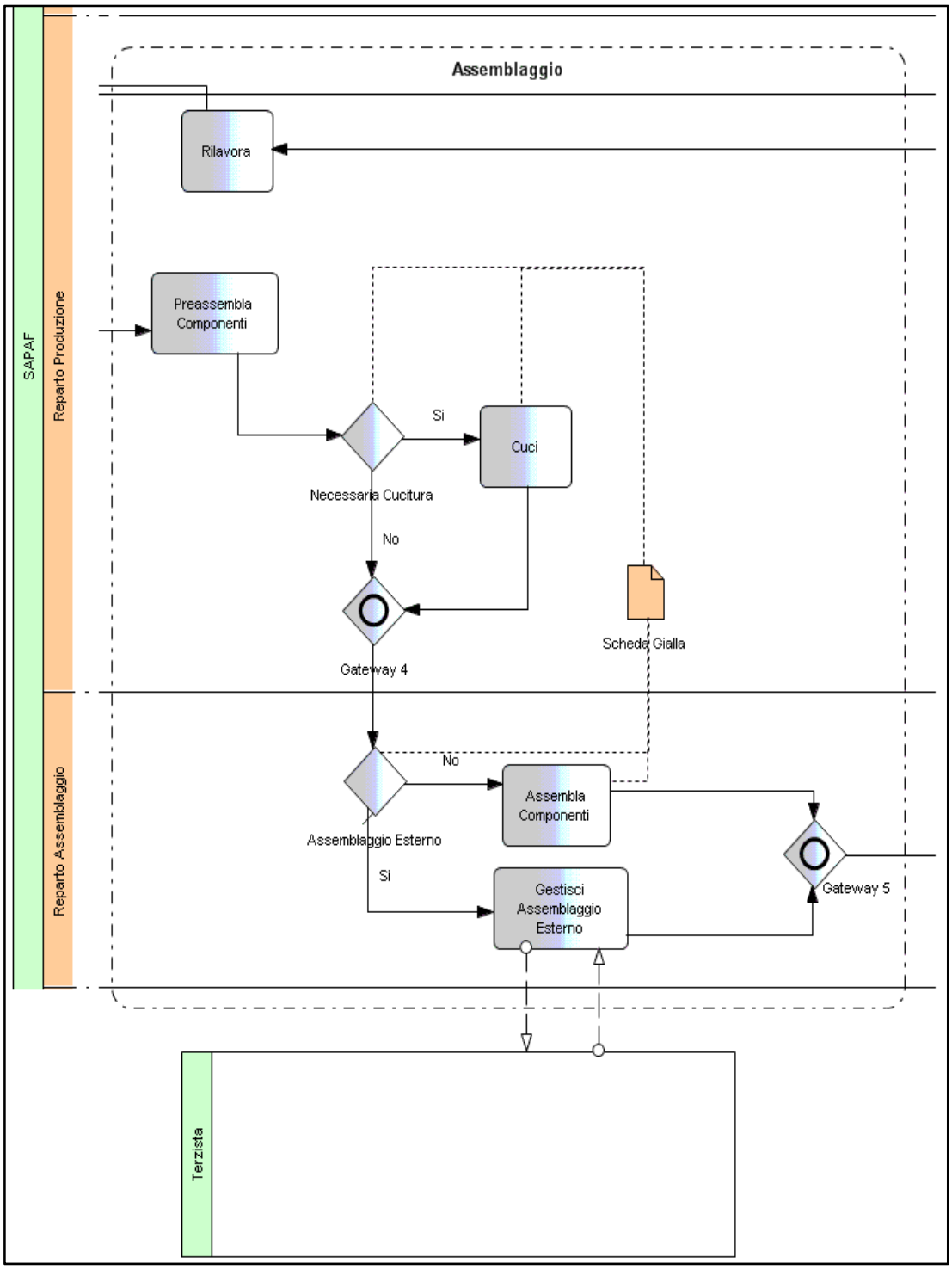
III-2. BPMN Top Level 2/6



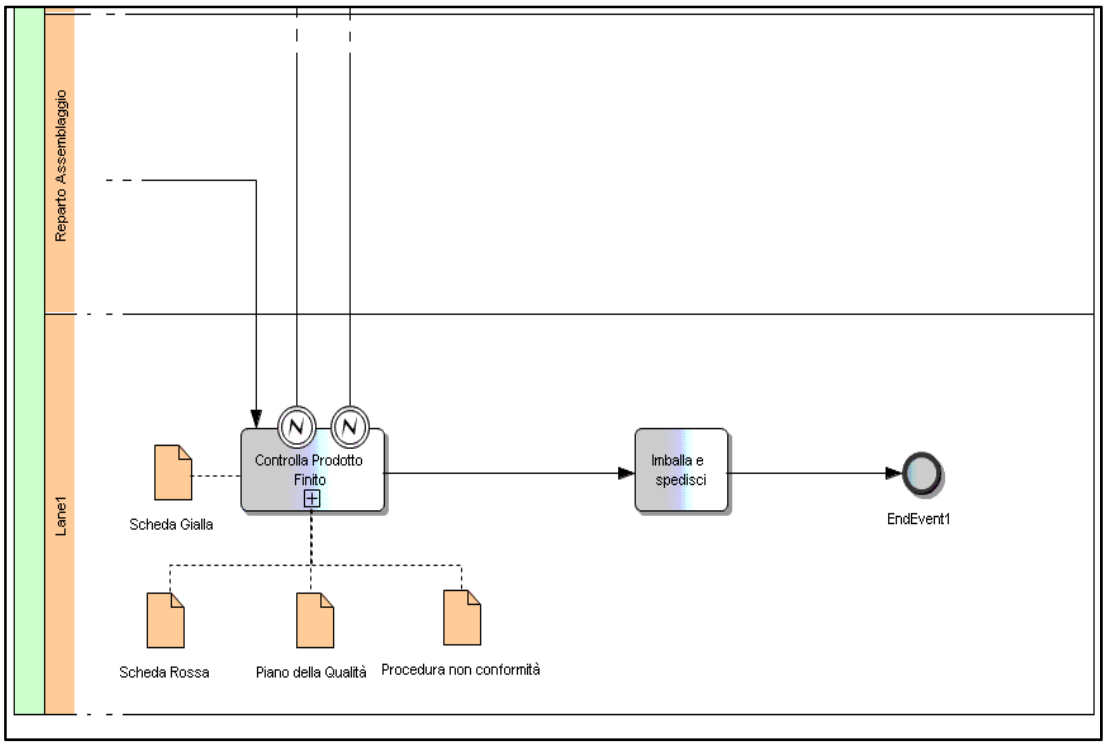
III-3. BPMN Top Level 3/6



III-4. BPMN Top Level 4/6

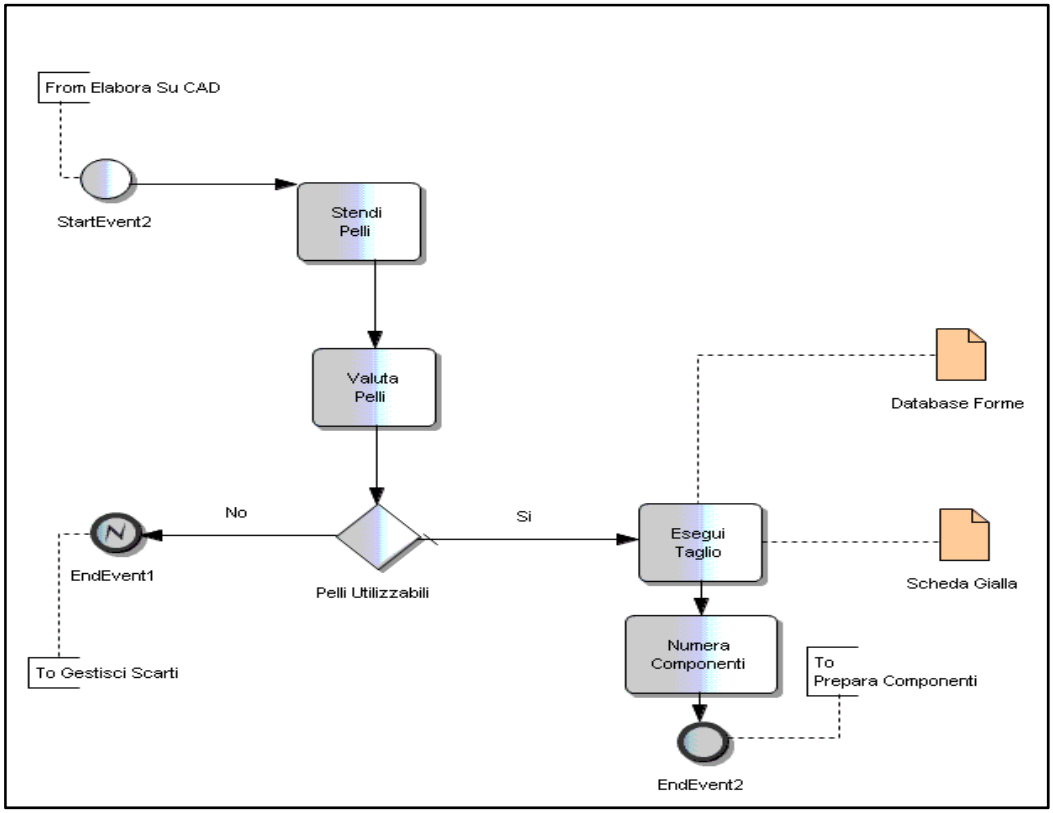


III-5. BPMN Top Level 5/6

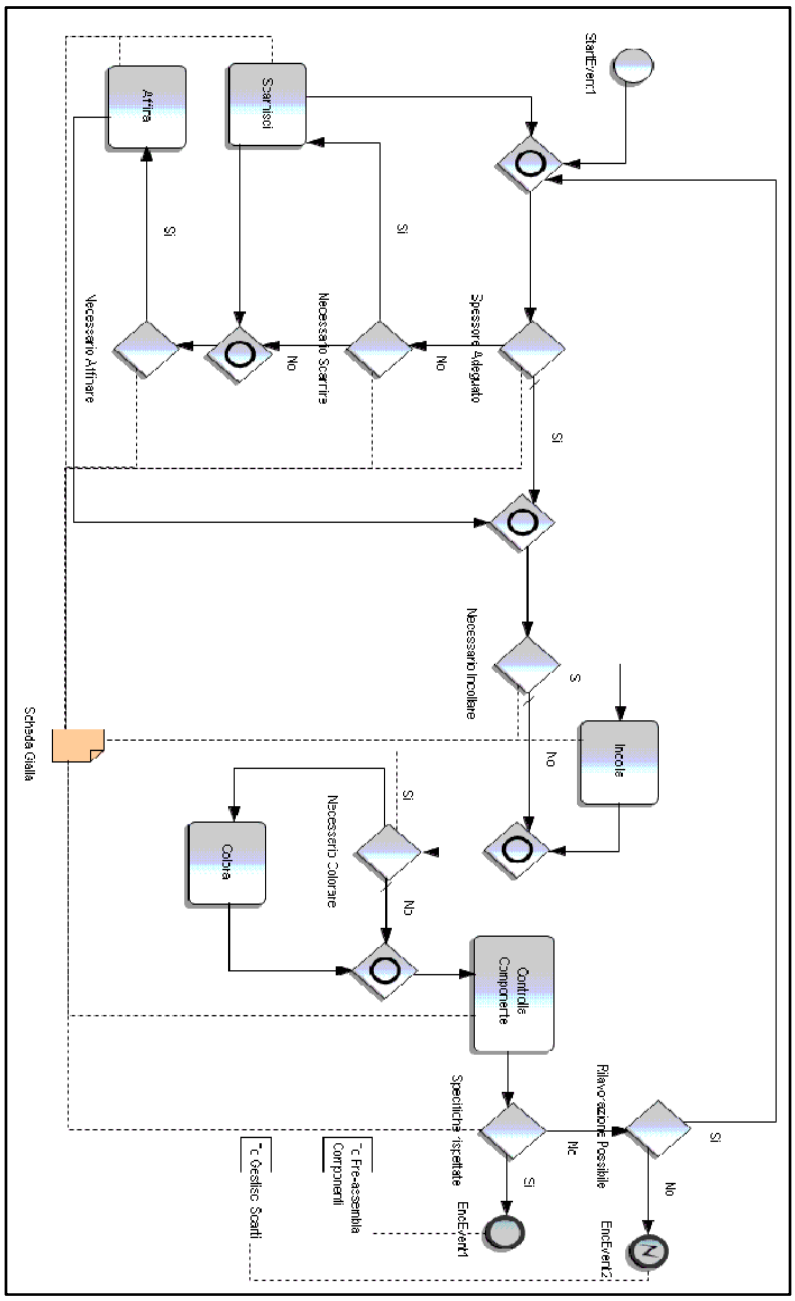


III-6. BPMN Top Level 6/6

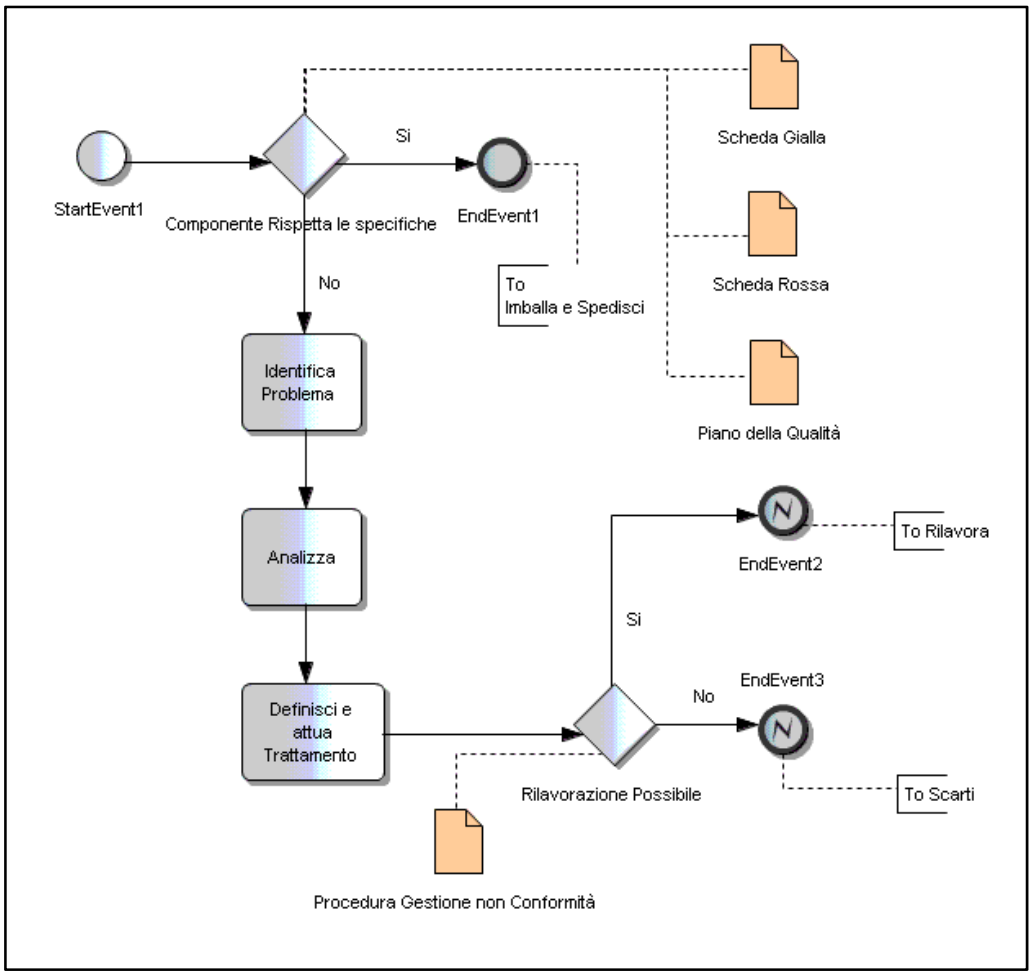
Riportiamo ora i diagrammi relativi ai sottoprocessi modellati nel Top Level.



III-7. BPMN Taglia



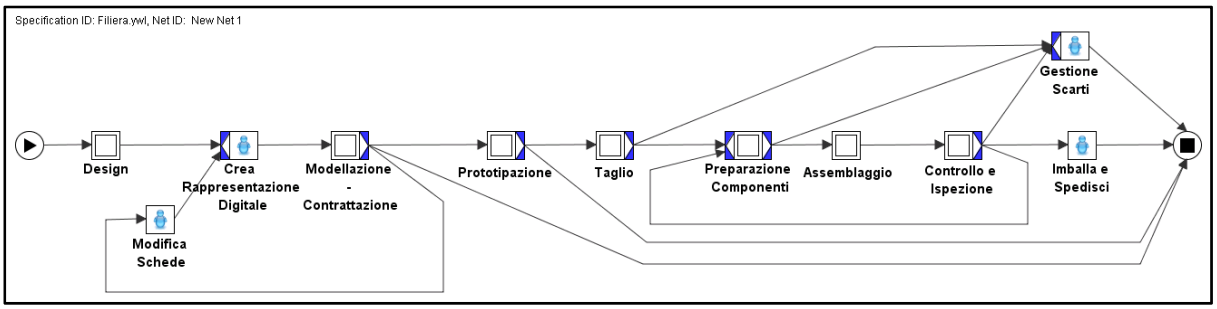
III-8. BPMN Prepara Componenti



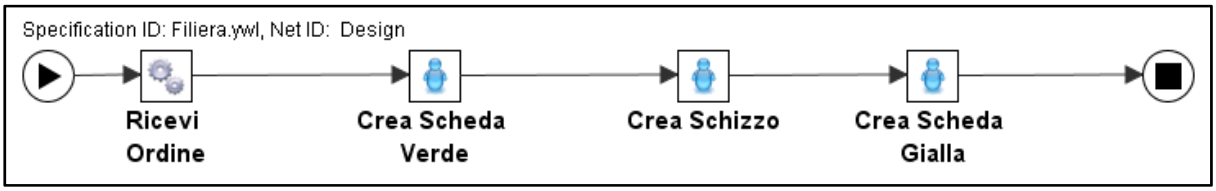
III-9. BPMN Controlla prodotto finito

Capitolo 19. Mappatura dei processi con YAWL

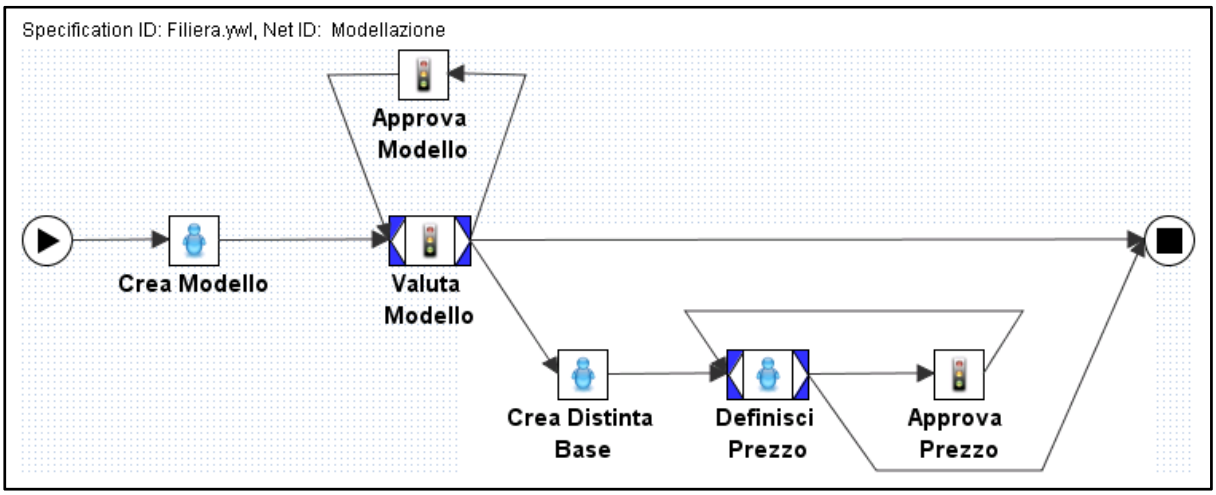
Riportiamo qui di seguito i diagrammi che modellano i processi del ciclo produttivo di una borsa all'interno della S.A.P.A.F, così come precedentemente descritti nel Capitolo 17.



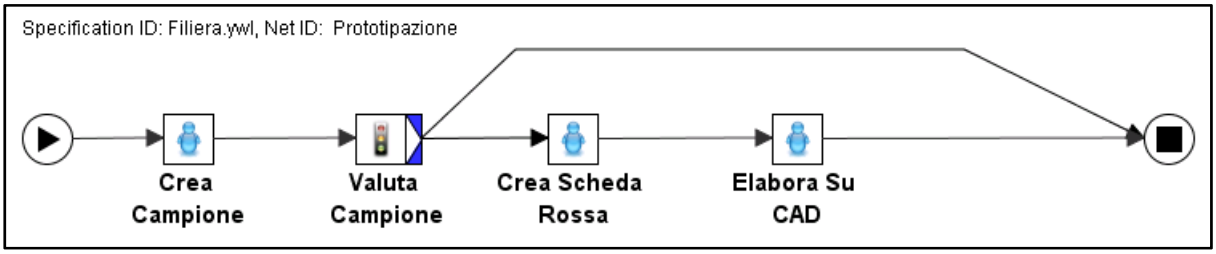
III-10. YAWL Top Level



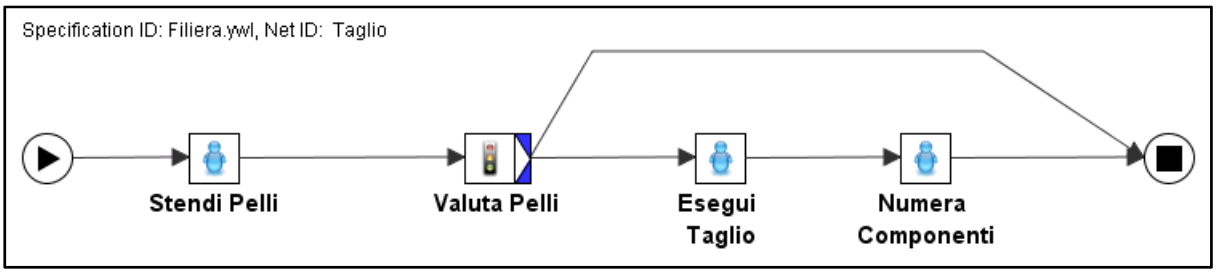
III-11. YAWL Design



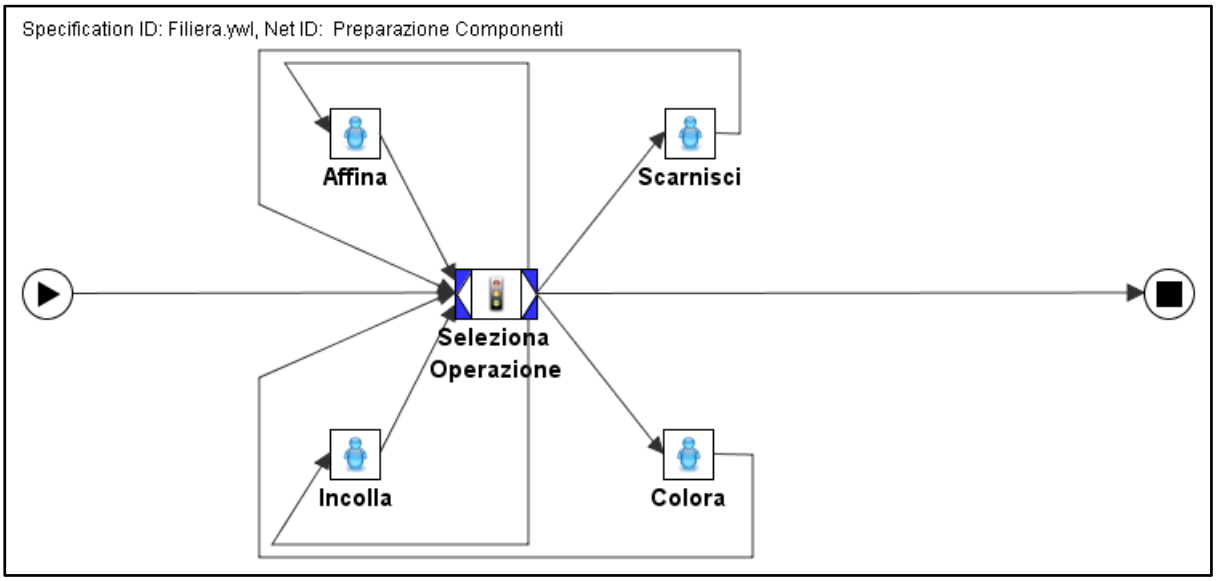
III-12. YAWL Modellazione – Contrattazione



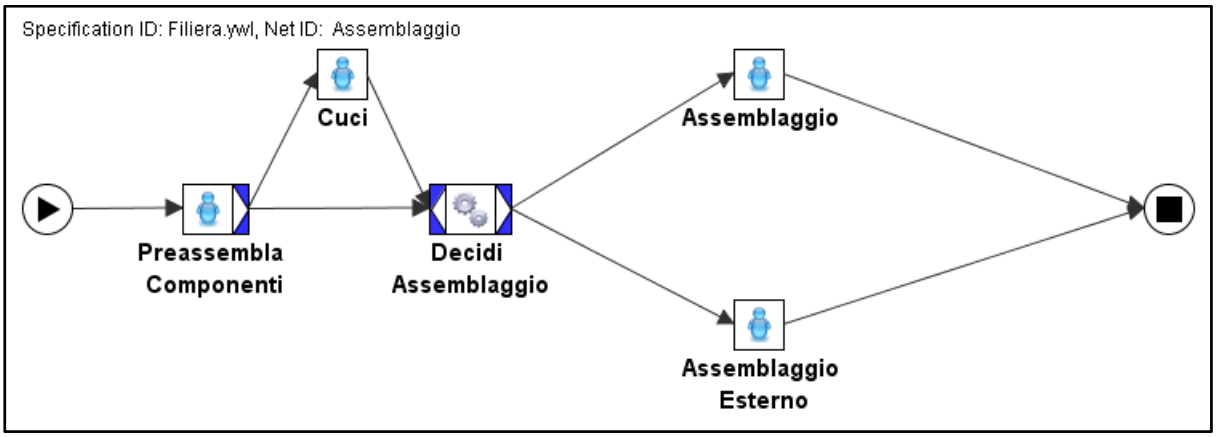
III-13. YAWL Prototipazione



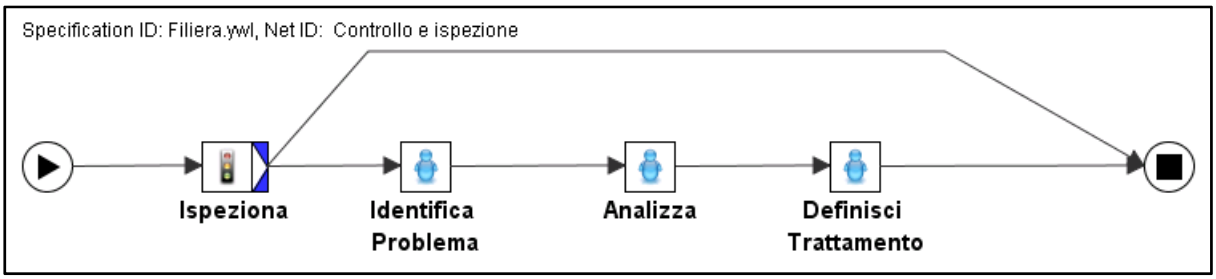
III-14. YAWL Taglio



III-15. YAWL Preparazione Componenti



III-16. YAWL Assemblaggio



III-17. YAWL Controllo e Ispezione

Capitolo 20. Confronto sul design

Nei capitoli precedenti abbiamo presentato i diagrammi che scaturiscono dagli approcci BPMN (1) e YAWL (22), in questo capitolo cercheremo di focalizzare l'attenzione sui pro e i contro dei due formalismi.

Semplicità di rappresentazione

Se si affiancano i diagrammi ottenuti dalla mappatura dei processi, la prima grossa differenza che salta all'occhio è la maggiore semplicità con cui YAWL (22) permette di delinearne il flusso di esecuzione. Come era facilmente intuibile già dalla presentazione del linguaggio, l'approccio attraverso le reti di Petri (65) permette una maggiore espressività e di conseguenza situazioni come cicli e controlli successivi diventano facilmente progettabili attraverso l'uso di transizioni condizionali invece che di gateway (utilizzati dal BPMN (1)). Per avere un rapido confronto basti pensare alla progettazione del sotto processo Preparazione dei Componenti mostrato nelle figure III-8 e III-15. Mentre nel primo caso è necessario utilizzare diversi gateway per la scelta dell'operazione necessaria alla preparazione del componente, in YAWL (22) questo avviene semplicemente permettendo la selezione dell'operazione, rendendo il diagramma più snello e comprensibile.

Maggiore controllo su dati, flusso e responsabilità

Il formalismo più rigoroso dell'approccio proposto da YAWL (22) permette un maggior controllo sia sulla coerenza dei dati che sulla correttezza del diagramma progettato. Come già visto infatti, la Data Perspective di YAWL (63) prevede la dichiarazione dello scambio di informazioni tra le varie net e i vari task, nonché la definizione delle condizioni di attivazione dei vari percorsi del flusso di controllo. In questo modo il progettista è obbligato a prevedere quali dati saranno necessari all'esecuzione del processo, di che tipo saranno (ricordiamo la tipizzazione attraverso XML Schema (16) della Data Perspective) e come deve avvenire lo scambio di informazioni tra i vari task. Le utilità di verifica e validazione della specifica offerte dall'ambiente di progettazione di YAWL (68), permettono di fare un'analisi statica sulla presenza di situazioni anomale dovute ad errori umani progettuali quali

- Incongruenza dei dati
- Insufficienza di dati
- Cicli infiniti
- Task non raggiungibili

La gestione delle risorse inoltre permette di modellare l'accesso ai vari task in esecuzione da parte del corretto utilizzatore sempre all'interno della stessa specifica.

Tale formalismo, non presente in BPMN (1), risulta quindi essere di notevole importanza nella progettazione dei processi.

Minore Espressività sui dati e sulle responsabilità

La notazione grafica di YAWL (22) risulta comunque essere un po' troppo scarna e soffre di una minore espressività per quanto riguarda le responsabilità e la necessità di dati per l'esecuzione dei Task. Nonostante infatti sia richiesto un maggior formalismo nella dichiarazione e definizione delle variabili di net e di task, questo non viene rappresentato nel diagramma, e sono necessarie delle note integrative per esplicitare tali caratteristiche del processo all'utilizzatore del diagramma stesso. In BPMN (1) invece, la necessità di documenti e risorse è immediatamente visibile grazie alla presenza degli artifact associati alle varie attività. Lo stesso discorso vale per quanto riguarda le responsabilità. Attraverso l'utilizzo di pool e lane, BPMN (1) permette di esplicitare già nel diagramma chi deve fare che cosa, e questo rende maggiormente comprensibile il diagramma stesso. Lo stesso costrutto dell'approccio BPMN (1) mette in risalto quali operazioni sono gestite in outsourcing (figura III-5 sulla fase di assemblaggio nel caso di operazione demandata a società terzista) e quali operazioni richiedono l'intervento del cliente (figure III-1 III-2 rispettivamente su fase di design e prima prototipazione) per quanto riguarda l'elicitazione dei requisiti, l'approvazione del prototipo della borsa e la contrattazione sul prezzo. Se andiamo a confrontare queste figure con i diagrammi III-16 III-11 III-12 (rispettivamente Assemblaggio, Design, Modellazione e Contrattazione) della mappatura in YAWL (22), salta infatti subito all'occhio la mancanza di una esaltazione del rapporto dell'azienda con terzisti e clienti, fasi di importanza cruciale per il Business Process Management che per questo non possono permettere ambiguità.

Capitolo 21. La traduzione in BPEL

Per poter vedere in esecuzione il processo mappato in BPMN (1), come già affrontato precedentemente in questa tesi, dobbiamo prima tradurre il diagramma in BPEL (5). Nonostante i vari vendor pubblicizzino a gran voce la potenza di traduzione BPMN (1)/BPEL (5) dei loro tools, nessuno di questi ultimi è in grado di tradurre pienamente il primo linguaggio nel secondo, data la diversa natura dei costrutti che essi supportano.

La teoria sulla rappresentatività dei linguaggi di modellazione (78)permette di fare un confronto diretto tra quali sono le potenzialità di BPEL (5) e BPMN (1) in modo da evidenziare quelle che sono le maggiori differenze tra i due. Il processo di traduzione risulta di conseguenza non banale e non automatizzabile per buona parte dei processi rappresentati in BPMN (1). La traduzione automatica può dare luogo a perdita di informazione e quindi necessita di ulteriore lavoro di rimodellazione del documento BPEL (5) ottenuto, aumentando il disaccoppiamento dei due modelli e lo sforzo necessario a creare una completa implementazione di una SOA. Di seguito viene riportato parte del lavoro di J. Recker e J. Mendeling (rispettivamente della Queensland University of Technology e della Vienna University of Economics and Business Administration) “On the translation between BPMN e BPEL: Conceptual Mismatch between Process Modeling Languages” (79) dove vengono evidenziate le discrepanze dell’accoppiamento BPMN (1) /BPEL (5).

Discrepanze concettuali tra BPMN e BPEL

L’oggetto principale del lavoro di Recker e Mendeling è stato applicare due approcci di valutazione consolidati e strutturati (*Representation Theory* (78) e *Workflow Patterns Framework* (60)) sia al BPEL (5) che al BPMN e mettere i risultati a confronto in modo da poter effettuare un’ulteriore analisi comparata riguardo alle strategie possibili per la traduzione da un linguaggio all’altro. E’ infatti noto che il Business Process Management ha diversi utilizzatori (architetto, analista, implementatore etc.), ognuno dei quali vede i processi sotto un’ottica diversa. Ne consegue che ognuno di questi avrà diverse priorità sulla necessità di rappresentare determinati aspetti del processo stesso (gerarchia delle attività, tipizzazione più o meno blanda degli oggetti, dettagli del flusso di controllo etc.). BPMN (1) e BPEL (5) nascono per supportare ottiche differenti, e quindi è naturale che abbiano una diversa capacità di rappresentazione. Di seguito verrà analizzato più nel dettaglio il risultato dell’analisi effettuata da Racker e Mandeling.

Representation Theory

BWW Construct	BPMN	BPEL
Things, including their Types and Properties		
THING	++	–
PROPERTY	N/A	N/A
in general	++	+
in particular	–	–
hereditary	–	–
emergent	–	+
intrinsic	–	–
mutual: non-binding	–	–
mutual: binding	–	–
attributes	–	+
CLASS	++	+
KIND	+	–
States assumed by Things		
STATE	–	+
CONCEIVABLE STATE SPACE	–	–
LAWFUL STATE SPACE	–	–
STATE LAW	–	–
STABLE STATE	–	–
UNSTABLE STATE	–	–
HISTORY	–	–

III-18. Representation Theory 1/2

Partendo dall'assunzione che la mappatura dei processi deve essere una rappresentazione della realtà, se vogliamo che esso possa davvero essere un portatore di valore e qualità per l'azienda, il modello utilizzato nella representation theory (78) (BWW model (80)) racchiude i costrutti di rappresentazione di oggetti e loro proprietà, in particolare: oggetti strutturati e tipizzati, stati ad essi relativi, eventi e transizioni ad essi correlati.

Events and Transformations occurring on Things		
EVENT	++	++
CONCEIVABLE EVENT SPACE	–	–
LAWFUL EVENT SPACE	–	–
EXTERNAL EVENT	++	+
INTERNAL EVENT	++	++
WELL-DEFINED EVENT	++	+
POORLY-DEFINED EVENT	++	++
TRANSFORMATION	++	++
LAWFUL TRANSFORMATION	++	++
stability condition	++	+
corrective action	++	–
ACTS ON	+	+
COUPLING	+	+
Systems structured around Things		
SYSTEM	++	+
SYSTEM COMPOSITION	++	+
SYSTEM DECOMPOSITION	++	–
SYSTEM STRUCTURE	–	+
SYSTEM ENVIRONMENT	++	–
SUBSYSTEM	++	–
LEVEL STRUCTURE	++	–

III-19. Representation Theory 2/2

Nelle tabelle mostrate in figura III-18 e figura III-19 viene indicato con "+" il caso in cui il linguaggio fornisca un costrutto per la rappresentazione di una caratteristica del modello BWW (80), "-" se tale caratteristica non è supportata e "++" se il linguaggio fornisce metodi

diversi di rappresentazione. Dalla tabella si può notare come vi siano notevoli differenze di rappresentazione tra i due linguaggi.

Per quanto riguarda la rappresentazione degli oggetti, mentre il BPMN (1) supporta classi, oggetti e tipi di oggetto, il BPEL (5) utilizza solo le classi, di conseguenza nella traduzione perderemmo ad esempio l'attenzione su una particolare istanza di oggetto laddove se ne necessiti la differenziazione e dovremmo ovviare a tale problema creando una classe apposta. D'altro canto il BPEL (5) permette una maggiore rappresentazione delle proprietà degli oggetti, rendendo impossibile tale gestione con un BPD (7).

Possiamo notare come entrambi i linguaggi presentino difficoltà di rappresentazione dello stato degli oggetti. Mentre questo non è un problema nell'ambito della traduzione, rimane comunque essere una grave mancanza dell'approccio supportato dall'OMG (4) Group nell'ottica della gestione dei processi.

Per quanto riguarda i cambiamenti di stato, e le transizioni tra eventi, possiamo notare come BPMN (1) offra una notevole quantità di costrutti di rappresentazione, diversamente da BPEL (5). Questo si traduce spesso nella necessità di accorpare eventi esterni in un unico evento (come verrà presentato più avanti nell'analisi del nostro caso di studio). Il BPEL (5) offre però un buon numero di costrutti per la differenziazione delle transizioni, il che porta di nuovo alla necessità di allegare commenti al BPD (7) (utili a chi poi creerà il documento BPEL (5)) in modo da poter permettere una mappatura più completa del processo.

E' infine evidente che il BPMN (1) risulti maggiormente espressivo per quanto riguarda la rappresentazioni dei sistemi. Questa caratteristica rende difficile il passaggio tra pool e lane e partner link, causando la necessità di un'ulteriore sforzo di rimodellazione del BPEL (5).

Workflow Patterns Framework

Il workflow pattern framework (60), è già stato ampiamente discusso nell'ambito di questa tesi, quindi ci limitiamo a riportare il confronto sulla capacità di rappresentazione dei due linguaggi presi in esame in questo capitolo.

Workflow Patterns	BPMN	BPEL	Workflow Patterns (ctd.)	BPMN	BPEL
Basic Control Flow			11. Implicit Termination	+	+
1. Sequence	+	+	*Multiple Instances Patterns*		
2. Parallel Split	+	+	12. MI without Synchronization	+	+
3. Synchronization	+	+	13. MI with a priori Design Time Knowledge	+	+
4. Exclusive Choice	+	+	14. MI with a priori Runtime Knowledge	+	–
5. Simple Merge	+	+	15. MI without a priori Runtime Knowledge	–	–
Adv. Synchronization			*State-Based Patterns*		
6. Multiple Choice	+	+	16. Deferred Choice	+	+
7. Synchronizing Merge	+/–	+	17. Interleaved Parallel Routing	+/–	+/–
8. Multiple Merge	+	–	18. Milestone	–	–
9. Discriminator	+	–	*Cancellation Patterns*		
Structural Patterns			19. Cancel Activity	+	+
10. Arbitrary Cycles	+	–	20. Cancel Case	+	+

III-20. Workflow patterns

In tabella viene riportato con "+" il supporto diretto del linguaggio al pattern, con "-" la mancanza di tale supporto e con "+/-" un supporto parziale.

Come è facile intuire, il problema maggiore è dato dal BPEL (5). Il suo ridotto supporto ai pattern (sub process, discriminator e merge in particolare) è uno dei maggiori problemi riportati nella grossa casistica di traduzioni BPMN (1)-BPEL (5). Non è infatti raro (anzi, si può quasi parlare di prassi) che l'intero BPD (7) vada rivisto e ristrutturato in modo da poter essere tradotto in BPEL (5).

Transformation Strategies

In base alle loro capacità di espressione e ai costrutti e approcci utilizzati per modellare la realtà possiamo definire BPMN (1) e BPEL (5) rispettivamente come linguaggi orientati ai grafi e ai processi. In questo paragrafo chiameremo **strutturato** un grafo i cui Split gateways corrispondono a join dello stesso tipo e i cui cicli iniziano e finiscono rispettivamente con XOR - Split e XOR - join. Indichiamo inoltre **aciclico** un grafo in cui nessun nodo può essere raggiunto da se stesso. Un processo BPEL (5) sarà invece **strutturato** se non presenta link.

Transformation Strategies from BPMN to BPEL	Structured BPMN	Acyclic BPMN	All BPMN	Transformation Strategies from BPEL to BPMN	Structured BPEL	All BPEL
Element-Preservation	-	+	-	Flattening	+	+
Element-Minimization	-	+	-	Hierarchy-Preservation	+	-
Structure-Identification	+	-	-	Hierarchy-Maximization	+	+
Structure-Maximization	+	+	-			

III-21. Transformation Strategies

Per quanto riguarda la trasformazione da BPMN (1) a BPEL (5) le strategie riportate sono

- Element Preservation : mappatura di ogni elemento del diagramma in un costrutto BPEL e unire tali elementi con dei link
- Element Minimization : partendo dalla strategia precedente vengono sostituite le attività vuote (come ad esempio i gateway) con dei link contenenti condizioni di transizione.
- Structure Identification : Approccio simile a quello proposto nella specifica di BPMN (1) che prevede di ridurre il grafo del BPD (7) ad una serie di costrutti strutturati in BPEL (5).
- Structure Maximization : Traduzione del BPD (7) in un insieme di attività strutturate e annidate.

Come predetto non esiste un approccio che sia in grado di trasformare un generico diagramma BPMN (1) in un documento BPEL (5).

Sulla trasformazione da BPEL (5) a BPMN (1) gli approcci sono:

- Flattening : Con questo approccio le attività strutturate del processo BPEL (5) vengono tradotte in una serie di gateways e archi non annidati.

- Hierarchy Preservation : Traduzione delle attività strutturate in sottoprocessi BPMN (1).
- Hierarchy Maximization : Traduzione del documento BPEL (5) in un sottoprocesso BPMN (1) in cui non esistano archi che ne attraversino i limiti.

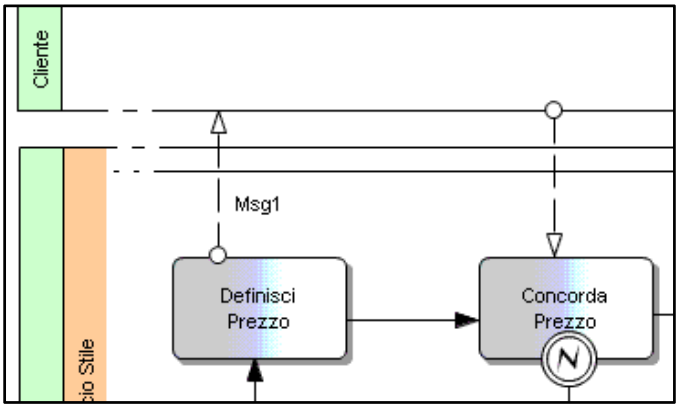
La traduzione da BPEL (5) a BPMN (1), che potremmo indicare come processo di reverse engineering, è possibile in ogni caso ma non tutte le tecniche permettono di mappare un generico documento BPEL (5) in un BPD (7).

Le problematiche esposte qui sopra fanno intuire che il tanto millantato processo automatico di traduzione in realtà non sia poi così immediato, ma al contrario richiede un notevole sforzo di rimodellazione ed adattamento del diagramma a quelli che sono i vincoli del BPEL (5). Ne consegue quindi un disaccoppiamento notevole tra processo progettato e processo implementato che riduce il controllo sul processo offerto al management.

Il BPEL (5) risulta essere infatti un linguaggio con dei vincoli espressivi ancora più stretti di quelli del BPMN (1), di conseguenza per poter tradurre un BPD (7) in BPEL (5) è necessario effettuare alcune modifiche al diagramma stesso. Analizziamo ora i principali problemi riscontrati nel processo di traduzione del nostro caso di studio.

Attività con send e receive

Mentre in BPMN (1) è possibile progettare attività che comunichino interattivamente con terze parti, in BPEL (5) questo non è consentito, quindi soluzione comune a tale problema è sdoppiare le attività in modo da separare l'inoltro e la ricezione dei messaggi. In figura III-22 viene riportato l'esempio dell'attività di definizione del prezzo precedentemente modellata con BPMN (1) in figura III-2.



III-22. Separazione di Send e receive

Processi incompleti

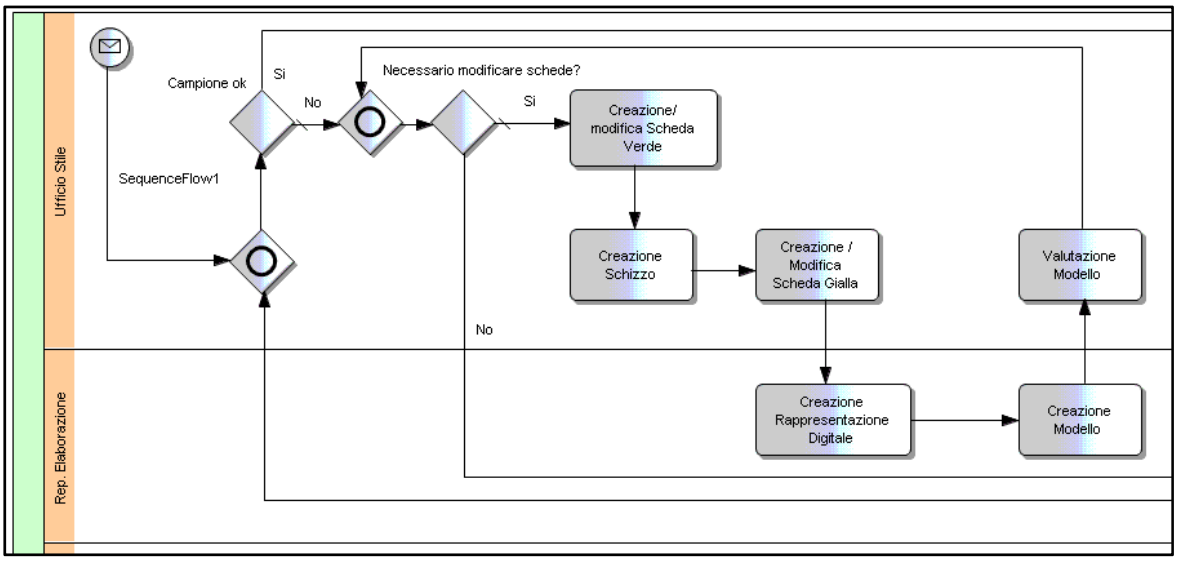
In BPMN (1) è possibile progettare dei processi incompleti pur rispettando la validità del modello. Nel nostro caso ad esempio il cliente (FIG III-1) è colui che da inizio al ciclo produttivo della borsa, ma le sue attività non sono completamente modellate nel BPD (7). Questo in BPEL (5) non è permesso, di conseguenza è stato necessario far diventare cliente e terzista delle black box.

Loop e merging

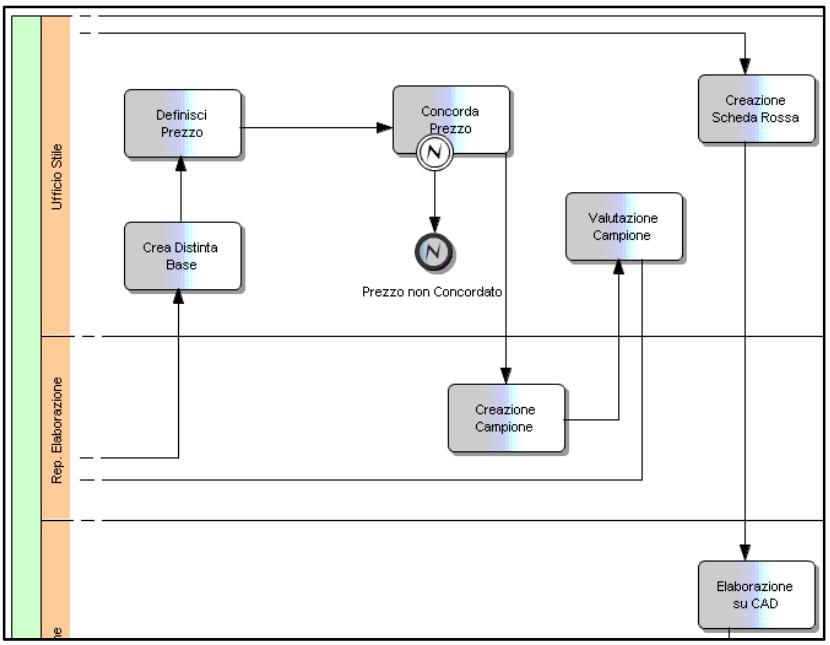
Il problema più significativo che si riscontra nel passaggio da BPMN (1) a BPEL (5) è la gestione dei cicli e dei merging. La libertà espressiva del BPD (7) infatti porta spesso ad un grafo non strutturato e ciclico e, come già affrontato (vedere FIG III-21), non vi è alcun possibile approccio di traduzione in questo caso.

BPD risultante

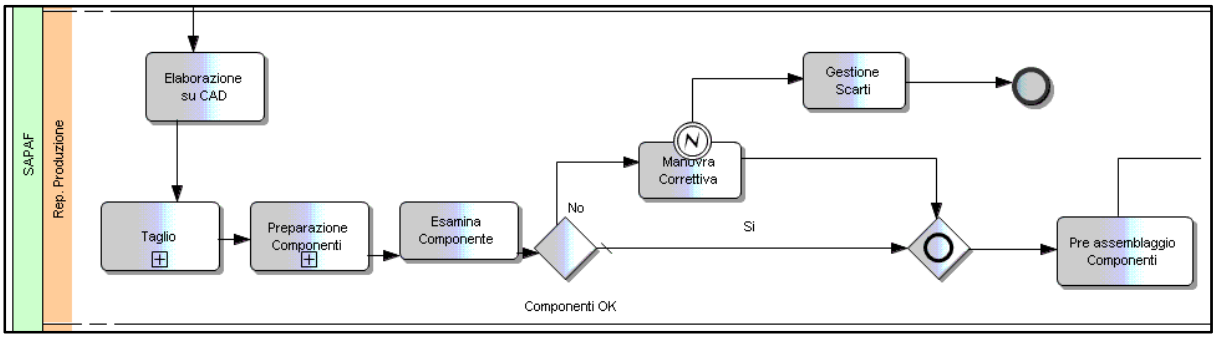
Di seguito viene riportato il diagramma BPMN (1) modificato in modo da poter essere tradotto in BPEL (5). Salterà subito all'occhio come il diagramma sia notevolmente cambiato rispetto a quello progettato precedentemente in base alla descrizione del processo in linguaggio naturale.



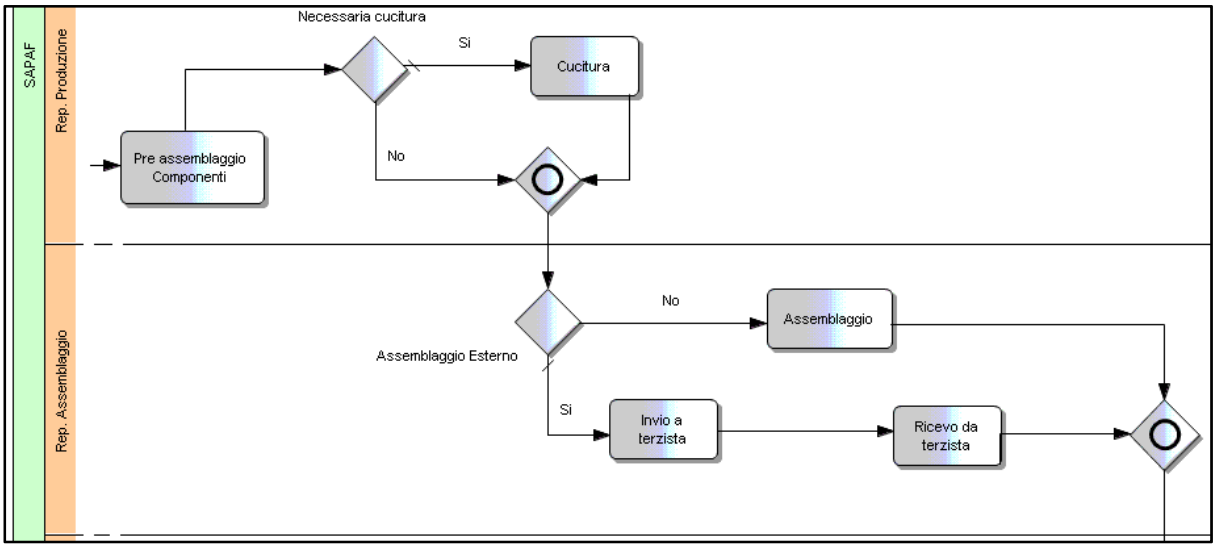
III-23. BPMN for BPEL Top level 1/5



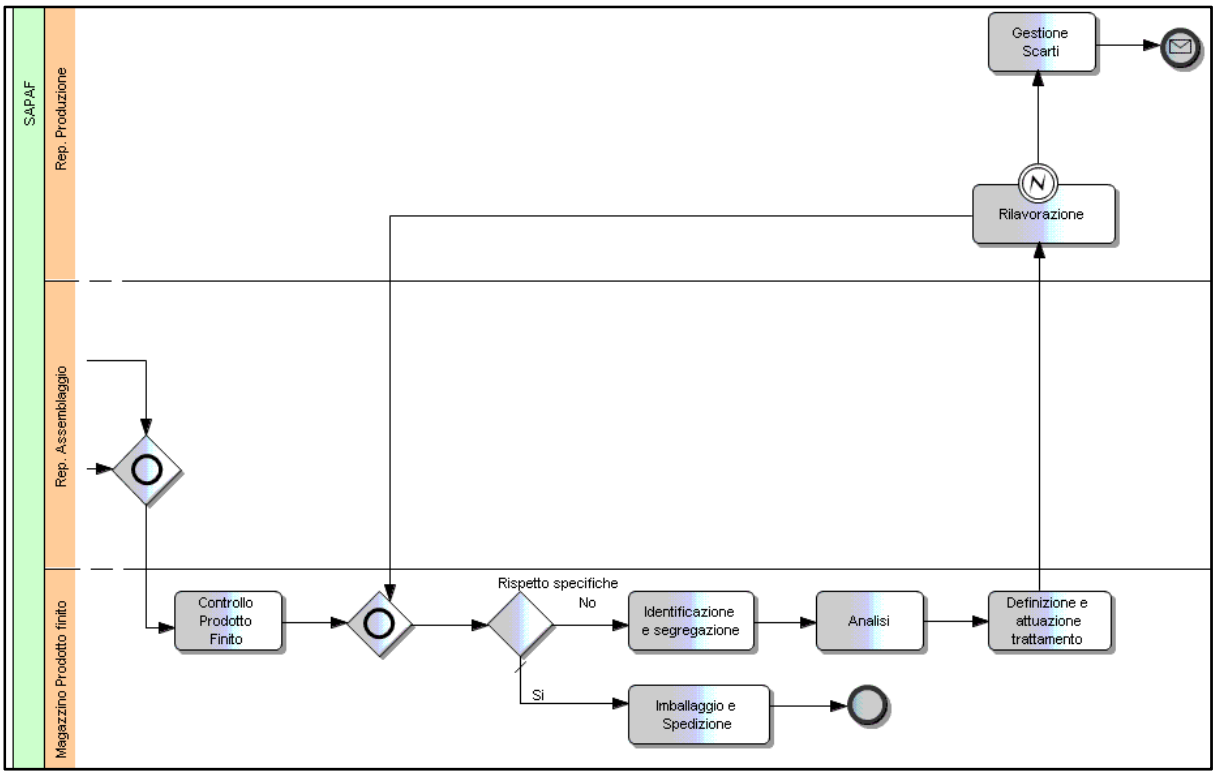
III-24. BPMN for BPEL Top level 2/5



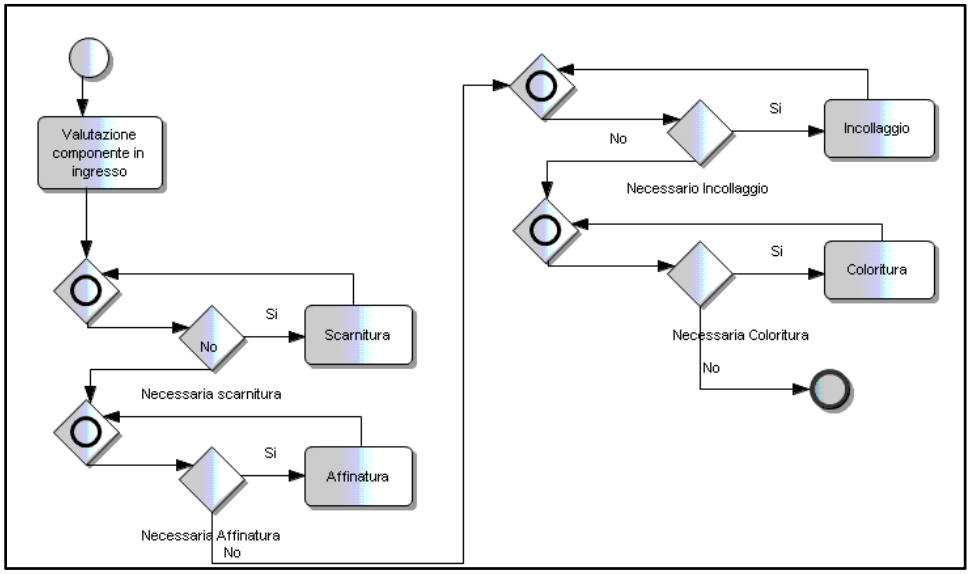
III-25. BPMN for BPEL Top level 3/5



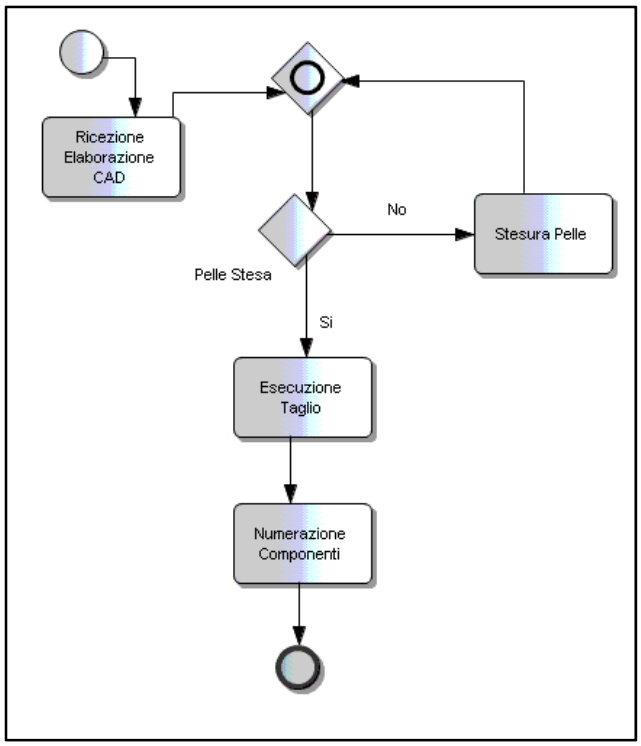
III-26. BPMN for BPEL Top level 4/5



III-27. BPMN for BPEL Top level 5/5



III-28. BPMN for BPEL Sottoprocesso Preparazione Componenti



III-29. BPMN for BPEL Sottoprocesso taglio

Artefatti necessari alla traduzione

Per poter effettuare una traduzione automatica da BPMN (1)a BPEL (5), uno volta adattato il BPD (7) ai requisiti per tale operazione come discusso precedentemente in questo capitolo, è necessario dichiarare e definire alcuni elementi tipici dei web server attraverso WSDL (67) (Web Service Description Language). Questo documento contiene la dichiarazione delle attività che compongono il processo da tradurre, definendone lo scambio di messaggi e i partecipanti a tali attività. WSDL (67) è un linguaggio derivante da XML (16)e la tipizzazione viene fatta con uno schema che può essere definita inline oppure importata da un documento XSD (XML Schema (18) Document). Lo sviluppo di questi documenti viene fortemente assistito dalla piattaforma ActiveVOS (21)Designer 6.0.2.0, rendendo più facile e snello l'oneroso lavoro di produrre codice XML (16). Di seguito riportiamo i documenti XSD (18) e WSDL (67).

BPEL Process

La produzione automatica di codice BPEL (5) permette di mappare il processo modellato con BPMN (1) (e adattato per poter essere tradotto) in un linguaggio eseguibile, ma come già affrontato (vedere "Discrepanze concettuali tra BPMN e BPEL" in questo stesso capitolo), BPEL (5) e BPMN (1) hanno una diversa espressività, di conseguenza il codice deve essere comunque ritoccato e maggiormente definito per poter essere validato. Viene riportato qui di seguito il codice BPEL completo e pronto per l'esecuzione.

Capitolo 22. Simulazione ed esecuzione del Processo BPEL

Nella programmazione è ben nota l'importanza delle attività di testing e debugging, ovvero quelle attività di controllo del codice prodotto che mirano rispettivamente a rilevare comportamenti inaspettati del programma e a ricercarne e rimuovere le cause correggendo le anomalie presenti nel codice (norma UNI EN ISO 9126 sulla gestione della qualità nei sistemi SW (81)).

Gli ambienti di sviluppo BPMN (1)/BPEL (5), in ottica di un'utenza prevalentemente di programmatori, nella maggior parte dei casi offrono il supporto a quella che viene chiamata **simulazione**. La simulazione viene fatta facendo girare il processo BPEL (5) su un server (solitamente integrato con l'ambiente di sviluppo stesso) fornendo come input dei sample data, ovvero insiemi di dati più o meno verosimili, realizzati dal programmatore in modo da verificare il rispetto delle specifiche di progetto da parte del processo realizzato.

La simulazione offre un ulteriore vantaggio rispetto al testing tradizionale, ovvero il confronto "*what-if*". E' infatti possibile a questo punto modificare il modello del processo per verificarne il comportamento nell'ottica di un cambiamento migliorativo del processo esecutivo vero e proprio, attraverso l'analisi di KPI (*Key Performance Indicator*) ovvero valori chiave definiti dall'analista del processo per valutarne il valore aggiunto. Tuttavia, abbiamo visto come il codice BPEL (5) prodotto si distacchi dal modello BPMN (1) realizzato dall'architetto del processo, riducendo così la possibilità dell'analisi "what - if".

Tale disaccoppiamento infatti:

- Aumenta il lavoro necessario per poter modificare il progetto a monte e produrre nuovamente il codice da simulare, rendendo meno appetibile l'idea di sfruttare questa possibilità
- Allontana la possibilità di simulare il diagramma realizzato dall'architetto del processo per la perdita di informazione legata al complesso processo di traduzione BPMN (1) /BPEL (5)

A fronte di tale situazione la tendenza è quella di operare direttamente sul codice BPEL (5) prodotto, allontanandolo ulteriormente dal BPD (7) originale traducendosi così in una perdita di qualità della mappatura del processo che si intende modellare.

Prima di poter eseguire una simulazione o l'esecuzione vera e propria sul server, dobbiamo ancora completare qualche passo realizzativo. Per potere fare il deploy di un processo BPEL (5) è infatti necessario:

- Realizzare un file dove definire le parti che entrano in causa nel progetto in termini di servizi web (.pdef)
- Realizzare descrittore del deploy, contenente informazioni utili al server per la gestione dei file annessi al progetto (descrittori, BPEL (5), wsdl (67), xsd (18) etc.) e per le impostazioni di sicurezza (.pdd)

Capitolo 23. Esecuzione del processo con YAWL

Essendo direttamente eseguibile, il processo modellato in YAWL (22) è pronto per essere caricato sul server e fatto girare. In questo modo il processo implementativo non subisce di perdita di informazione rispetto a quello progettuale, riducendo il lavoro umano necessario e permettendo di eseguire quello che viene modellato graficamente, in un ottica WYDIWYE (What You Design Is What You Execute).

Per quanto riguarda la struttura dati di interesse, abbiamo utilizzato lo schema xml generato con il supporto di ActiveVos (21) Designer 6.0.2.0, la cui interfaccia per questo scopo è molto più evoluta e riduce la possibilità di errore umano. A parte l'analisi statica del modello, non vi è però possibilità di fare prove, se non facendolo girare direttamente sul server attraverso l'interfaccia di amministrazione.

Come avviene per il BPEL (5), è necessario assegnare le risorse (ovvero gli elementi dell'organigramma dell'azienda) ad ogni Manual task, in modo da poter così essere assegnati. Una volta fatto partire l'Engine di YAWL (22), è possibile collegarsi dall'ambiente di progettazione per poter caricare i dati riguardanti l'organigramma aziendale in modo da utilizzare risorse già esistenti nel sistema.

Capitolo 24. Conclusioni

Mettendo a confronto l'approccio BPMN e l'approccio YAWL ne abbiamo analizzato le caratteristiche, valutandone le attitudini per offrire un ambiente di descrizione dei processi e di progettazione utilizzabile come strumento per far comunicare i diversi ruoli professionali coinvolti nella gestione dei processi. In particolare sono risaltate le carenze di BPMN, che pure si sta affermando come standard nel campo del BPM. La coppia BPMN / BPEL, anche se risulta essere molto potente, presenta aspetti ancora inadeguati per conciliare le esigenze dia dei progettisti che degli utenti di BPM.

Riassumendo, il BPMN (1) utilizza una notazione vicina ai progettisti del processo, ma non è eseguibile, il che introduce la necessità di una traduzione in un opportuno linguaggio eseguibile. La prassi consolidata è quella di seguire le indicazioni suggerite dallo standard BPMN (1) che presenta in appendice una parziale traduzione del BPMN (1) in BPEL (5). Questo linguaggio, nato per l'orchestrazione dei web services e orientato ad una programmazione molto strutturata, poco si adatta alla modellazione di realtà complesse quali i processi di business. Sebbene infatti la maggior parte dei tool presenti in commercio offrano la possibilità di una traduzione automatica del BPD in BPEL (5), questa è possibile solo per un ristretto sotto insieme dei processi realizzabili in BPMN (1). Ne consegue che il BPD realizzato dai progettisti debba essere ritoccato per permetterne una traduzione in BPEL (5). L'introduzione di questo ulteriore passaggio di rimodellazione presenta ovvi problemi di manutenibilità della documentazione relativa al processo di business. Altro problema nasce qualora si voglia effettuare un reenginering del processo attraverso delle analisi "what-if". Poiché le simulazioni debbono essere fatte in ambiente BPEL (5), si è indotti ad operare direttamente su quest'ultimo, correndo Il rischio di far eseguire un processo diverso da quello modellato ad alto livello e riducendo così l'efficienza e l'efficacia della gestione del processo.

La soluzione proposta da YAWL foundation (22) presenta aspetti di grande interesse, non ha ancora raggiunto le caratteristiche funzionali e la maturità necessarie per essere una reale alternativa all'approccio dell'OMG (4) Group, a causa della scarsità della notazione grafica (vedi Capitolo 20) e del fatto che l'ambiente di sviluppo e di esecuzione dei processi modellati con YAWL (63) non offre funzionalità pari a quelle attualmente presenti sul mercato. Come è infatti noto, lo sviluppo e commercio di strumenti basati su tecnologie standard, persegue da parte dei vari produttori la strategia di differenziazione che produce funzionalità aggiuntive vantaggiose per gli utenti. Ciò rappresenta è una notevole barriera all'entrata sul mercato per la YAWL foundation (22), il cui prodotto risulta essere meno soddisfacente per i manager.

L'applicazione dei due approcci a un caso di studio nell'ambito dei WFMS ha evidenziato le differenze concettuali presenti tra i vari linguaggi utilizzati nel BPM, che si riflettono nelle difficoltà incontrate nella progettazione ed esecuzione dei processi di business considerati. Al termine dell'esperienza svolta si individua l'esigenza di poter disporre di un unico ambiente, fruibile dai progettisti del processo, che permetta di definire processi da

analizzare e validare staticamente per i quali esaminare le opportune ottimizzazioni e infine da eseguire nello stesso contesto.

In questo modo infatti vengono eliminati alcuni passi intermedi tra la progettazione e l'esecuzione del processo, che possono essere dannosi per raggiungere gli obiettivi di qualità totale.

Un'interessante ulteriore analisi su questo aspetto potrebbe essere quella sul caso di Oracle (20) BPM Studio 10.3.1.0.0. Oltre all'interfaccia molto più orientata agli utenti del BPM più che alla progettazione, Oracle (20) offre una componente a valore aggiunto notevole: la simulazione del processo in BPMN (1) direttamente nell'ambiente di progettazione, con report immediati. Questo è possibile grazie all'utilizzo di un linguaggio proprietario chiamato PBL (Process Business Language (82)), sviluppato precedentemente da BEA Systems (11), che permette l'esecuzione diretta del BPD (7).

L'approccio di Oracle dimostra che c'è spazio per proseguire sulla linea proposta da YAWL foundation facendo raggiungere ai prodotti di WFMS una maggiore maturità tecnologica e funzionale.

Ringraziamenti

La realizzazione di questa tesi rappresenta il momento culminante di un lungo processo formativo professionale e personale. Vorrei quindi approfittare di questa occasione per ringraziare tutte le persone che mi hanno aiutato e accompagnato in questi anni e in particolar modo

I miei genitori e le mie sorelle, per aver sempre creduto in me ed avermi insegnato a fare altrettanto

I miei compagni d'arte, grazie ai quali posso rendere ogni giorno più vividi i colori di un sogno

I miei amici più stretti a Pisa e a Cagliari, per essere la mia seconda famiglia nella vita.

"Vorrei fare un brindisi a mia madre che ha trombato e ha fatto un figo come me"

Rupert Sciamenna , 2008

Bibliografia

1. BPMN official website. [Online] www.bpmn.org.

2. BPMN Specification. [Online] http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf.

3. BPMI official website. [Online] www.bpmi.org.

4. OMG official website. [Online] http://www.omg.org/.

5. BPEL 2.0 Specification. [Online] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.

6. BPEL4WS 1.1 Specification. [Online] http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

7. BPMN 1.1 Poster. [Online] http://bpt.hpi.uni-potsdam.de/pub/Public/BPMNCorner/BPMN1_1_Poster_EN.pdf.

8. W3C Web Service Activity Page. [Online] http://www.w3.org/2002/ws/.

9. IBM official website. [Online] www.ibm.com.

10. Microsoft official website. [Online] www.microsoft.com.

11. Information about the Oracle and BEA Systems acquisition. [Online] http://www.oracle.com/bea/index.html.

12. SAP official website. [Online] www.sap.com.

13. Information about the Oracle and Siebel Systems acquisition. [Online] http://www.oracle.com/siebel/index.html.

14. Oasis official website. [Online] http://www.oasis-open.org/.

15. Description on the upsoming changes from BPEL1.1 to BPEL 2.0. [Online] http://webservices.sys-con.com/read/155617.htm.

16. W3C XML web page. [Online] http://www.w3.org/XML/.

17. WSDL 1.1 Specification. [Online] http://www.w3.org/TR/wsdl.

18. W3C XML Schema page. [Online] http://www.w3.org/XML/Schema#dev.

19. Intalio official website. [Online] www.intalio.com.

20. Oracle official website. [Online] www.oracle.com.

21. ActiveVOS official website. [Online] www.activevos.com.

22. YAWL official website. [Online] www.yawl-system.com.

23. Eclarus official website. [Online] www.eclarus.com.

24. RedHat official website. [Online] www.redhat.com/.

25. SUSE official website. [Online] http://www.novell.com/linux/suse/.

26. MAC OSX official website. [Online] http://www.apple.com/macosx/.

27. Sun Solaris official website. [Online] http://www.sun.com/software/solaris/.

28. HP-UX official website. [Online] http://www.hp.com/go/hpux/.

29. Apache Tomcat official website. [Online] http://jakarta.apache.org/tomcat/.

30. JBoss official website. [Online] http://www.jboss.org/.

31. Apache Geronimo official website. [Online] http://geronimo.apache.org/.

32. MySQL official website. [Online] www.mysql.com.

33. PostgreSQL official website. [Online] http://www.postgresql.org/.

34. Sybase official website. [Online] www.sybase.com.

35. Apache Derby official website. [Online] http://db.apache.org/derby/.

36. Aris Official Website. [Online] http://www.aris.com/.

37. WS-Human Task Specification. [Online] http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf.

38. BPEL4People Specification. [Online] http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_v1.pdf.

39. W3C XPath page. [Online] http://www.w3.org/TR/xpath.

40. W3C XQuery page. [Online] http://www.w3.org/TR/xquery/.

41. W3C XForms page. [Online] http://www.w3.org/TR/xforms/.

42. W3C XSL page. [Online] http://www.w3.org/TR/xslt.

43. W3C SOAP page. [Online] http://www.w3.org/TR/soap/.

44. RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1. [Online] http://www.ietf.org/rfc/rfc2616.txt.

45. RFC2818: HTTPS. [Online] http://www.ietf.org/rfc/rfc2818.txt.

46. JMS Specification. [Online] http://dlc.sun.com/pdf/816-5904-10/816-5904-10.pdf.

47. **Fielding, Roy.** Representational State Transfer (REST). [Online] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

48. Java official website. [Online] http://www.java.com/.

49. WS-Addressing Specification. [Online] http://www-128.ibm.com/developerworks/library/specification/ws-add/.

50. WS-Reliable Messaging Specification. [Online] http://www.ibm.com/developerworks/library/specification/ws-rm/.

51. WS-I Basic Profile Specification. [Online] http://www.ws-i.org/Profiles/BasicProfile-1.0.html.

52. WS-Security Specification. [Online] http://www-128.ibm.com/developerworks/library/specification/ws-secure/.

53. SAML Specification. [Online] http://saml.xml.org/saml-specifications.

54. LDAP Specification roadmap. [Online] http://tools.ietf.org/html/rfc4510.

55. WS- Policy Specification. [Online] http://schemas.xmlsoap.org/ws/2004/09/policy.

56. UDDI Specification. [Online] http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm.

57. WSIL Specification. [Online] http://www-128.ibm.com/developerworks/library/specification/ws-wsilspec/.

58. WS-Distributed Management Specification. [Online] http://www.oasis-open.org/specs/#wsdmv1.1.

59. Eclipse official website. [Online] http://www.eclipse.org/.

60. *Workflow Patterns Initiative official website.* [Online] http://www.workflowpatterns.com/.

61. Eindhoven University of Technology official website. [Online] www.tue.nl.

62. QUT | Queensland University of Technology official website. [Online] www.qut.edu.au/.

63. **Hofstede, W.M.P. van der Aalst and A.H.M. ter.** YAWL: Yet Another Workflow Language. [Online] http://eprints.qut.edu.au/archive/00010244/01/10244.pdf.

64. **Kurt Jensen, , Lars M. Kristensen.** *Coloured Petri Nets: Modeling and Validation of Concurrent Systems (1st Edition).* s.l. : Springer, 2009. ISBN 3-540-62867-3.

65. Petri Nets World official website. [Online] http://www.informatik.uni-hamburg.de/TGI/PetriNets/.

66. LGPL official website. [Online] http://www.gnu.org/copyleft/lesser.html.

67. W3C WSDL page. [Online] http://www.w3.org/TR/wsdl.

68. **Moe T. Wynn, H.M.W. Verbeek, Wil M.P. van der Aalst, Arthur H.M. ter Hofstede, and David Edmond.** Business Process Verification - Finally a Reality! . [Online] http://eprints.qut.edu.au/archive/00009107/01/9107.pdf.

69. **W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede.** Design and Implementation of the YAWL system. [Online] http://eprints.qut.edu.au/archive/00000379/01/aalst_yawls.pdf.

70. Apache WSIF official website. [Online] http://ws.apache.org/wsif/.

71. URL RFC Document. [Online] http://www.w3.org/Addressing/rfc1738.txt.

72. **Michael Adams, Arthur H. M. ter Hofstede, David Edmond.** Implementing Dynamic Flexibility in Workflows. [Online] http://www.yawl-system.com/documents/Implementing%20Worklets.pdf.

73. **Nick Russel, Arthur H.M. ter Hofstede and Wil M.P. van der Aalst.** newYAWL: Specifying a Workflow Reference Language using Coloured Petri Nets. [Online] http://www.yawl-system.com/documents/newYAWL-cpn.pdf.

74. **Stefani, Ing. Davide.** Modellazione BPMN per piattaforme di Business Process management: esperienze nel settore della pelletteria. *Tesi di laurea.* Università di Pisa, facoltà di Ingegneria : s.n., 2008.

75. **Mario Giovanni C.A. Cimino, Francesco Marcelloni.** *INNO.PRO.MODA: innovazione, progettazione, qualità e tracciabilità per il sistema moda.* s.l. : Pacini Editore, 2008.

76. REGIONE TOSCANA - RICERCA INDUSTRIALE TRASFERIMENTO INNOVAZIONE sito web ufficiale. [Online] www.innovazione.toscana.it.

77. PMI.it - informazione e tecnologia per piccole e medie imprese. [Online] www.pmi.it.

78. Rappresentation Theory journal official website. [Online] http://www.ams.org/ert/.

79. **Jan Recker, Jan Mendling.** On the Translation between BPMN and BPEL:Conceptual Mismatch between Process Modeling Languages. [Online] http://wi.wu-wien.ac.at/home/mendling/publications/06-EMMSAD.pdf.

80. Information Systems Foundations: Theory, Representation and Reality. [Online] http://epress.anu.edu.au/info_systems02/mobile_devices/index.html.

81. International Standard Organization official website. [Online] www.iso.org.

82. Process Business Language (PBL) reference. [Online] http://download.oracle.com/docs/cd/E13154_01/bpm/docs65/studio/index.html?t=modules/process_business_language/c_Head_Process_Business_Language.html.